802.1AX™

IEEE Standard for Local and metropolitan area networks—

Link Aggregation

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

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

3 November 2008

IEEE Std 802.1AX™-2008

IEEE Std 802.1AX™-2008

IEEE Standard for
Local and metropolitan area networks—

Link Aggregation

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 26 September 2008

IEEE-SA Standards Board

Abstract: Link Aggregation allows one or more links to be aggregated together to form a Link Aggregation Group, such that a Media Access Control (MAC) Client can treat the Link Aggregation Group as if it were a single link. To this end, it specifies the establishment of data terminal equipment (DTE) to DTE logical links, consisting of N parallel instances of full duplex point-to-point links operating at the same data rate. This standard defines the MAC independent Link Aggregation capability, and general information relevant to specific MAC types that support Link Aggregation.
Keywords: Aggregated Link, Aggregator, Link Aggregation, Link Aggregation Group, local area network, management

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

PDF: ISBN 973-07381-5794-8 STD95818
Print: ISBN 973-07381-5795-5 STDPD95818

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 or the soundness of any judgments 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. When a document is more than five years old and has not been reaffirmed, 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 judgement 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. Comments on standards and requests for interpretations should be submitted to the following address:

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

Introduction

This introduction is not part of IEEE Std 802.1AX-2008, IEEE Standard for Local and Metropolitan Area Networks—Link Aggregation.

The majority of the content of this standard is from IEEE Std 802.3™ where it was originally developed as the amendment project IEEE P802.3ad. Management information is from Clause 30 with the bulk of the functionality from Clause 43 and its annexes.

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

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 amendment may require use of subject matter covered by patent rights. By publication of this amendment, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this amendment are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

The following individuals were officers and members of the IEEE 802.3 working group at the beginning of the working group ballot. Individuals may have not voted, voted for approval, disapproval or abstained on this standard.

Robert M. Grow, *IEEE 802.3 Working Group Chair, IEEE 802.3ax Task Force Editor*
David J. Law, *IEEE 802.3 Working Group Vice-Chair, IEEE 802.3ax Task Force Chair*
Wael William Diab, *IEEE 802.3 Working Group Secretary*
Steven B. Carlson, *IEEE 802.3 Working Group Executive Secretary*
Bradley Booth, *IEEE 802.3 Working Group Treasurer*

John Abbott
Joe Abler
Taufique Ahmed
Youichi Akasaka
Arne Alping
Michael Altman
Pete Anslow
Joseph N. Babanezhad
Jim Barnette
Hugh Barrass
Scott Barrick
Howard Baumer
Denis Beaudoin
Michaël Beck
Christian Beia
Michael Bennett
Ralf-Peter Braun
Dave Brearley
Matt Brown
Phillip Brownlee
Robert Brunner
Robert Busse
Jeffrey Cain
J. Martin Carroll
Mandeep Chadha
David Chalupsky
Frank Chang
Luke Chang
George Claseman
Terry Cobb
James Collum
David Cunningham
Fumio Daido
John Dallesasse
Yair Darshan
Piers Dawe
Bill Delveaux
Joe DeNicholas
Claudio DeSanti
Suveer Dhamejani
Chris Di Minico
Thomas Dineen
Thuyen Dinh
Dan Dove
Mike Dudek
Marcus Duelk
Joseph E. Dupuis
Frank J. Effenberger
Daniel Feldman
Thomas Fischer
Alan Flatman
Howard M. Frazier
Richard Frosch
Ilango S. Ganga
Ali Ghiasi
Sajol Ghoshal
Russ Gyurek
Marek Hajduczenia
Brian Holden
Keith Hopwood
Walter K. Hurwitz
John Jaeger
John Jetzt
Wenbin Jiang
Andrew C. Jimenez
Chad Jones
Thomas K. Jørgensen
Shinkyo Kaku
Boris Katzenberg
Tatsuya Kawashimo
Mitsunobu Kimura
David J. Koenen
Paul Kolesar
Derek Koonce
Glen Kramer
Jan Krellner

Hans Lackner
D. Matthew Landry
Daun Langston
Jeff Lapak
Ferdinando Lari
Dong-Soo Lee
Worayot Lertniphonphun
Robert Lingle, Jr.
Jeffrey Lynch
Eric R. Lynskey
Jeff Mandin
Arthur Marris
Thomas Mathey
Mike McConnell
Michael S. McCormack
Richard Y. Mei
Richard Mellitz
Tremont Miao
Charles Moore
Andy Moorwood
Wayne A. Mueller
Toshiaki Mukojima
Shimon Muller
Gary Nicholl
Shinji Nishimura
Ron Nordin
Bob Noseworthy
Mark Nowell
Steven O'Hara
Satoshi Obara
Gourgen Oganessyan
Bengt-Erik Olsson
Akihiro Otaka
George Oulundsen, III
Tom Palkert
Tim Parker
Glenn Parsons
Martin Patoka
Arkadiy Peker
Petar Pepeljugoski
John Picard
Petre Popescu
Carl R. Posthuma
Rick Rabinovich
Prakash Radhakrishnan
Duane Remein
Steve Robbins
Valerie Rybinski
Naoto Saeki
Ramesh Sastry
Olindo Savi
T. Shannon Sawyer
Fred Schindler
Peter Schoenmaker
Lee Sendelbach
Shoichiro Seno
Masayuki Shigematsu
Larry Shorthill
Andrew Smith
Fulvio Spagna
Clay Stanford
Ken-ichi Suzuki
Steve Swanson
Andre Szczepanek
Akira Takahashi
Noriyuki Takeda
Motoyuki Takizawa
Keiji Tanaka
Tsutomu Tatsuta
Vivek Telang
Geoffrey O. Thompson
Peter Tomaszewski
Hidehiro Toyoda
Stephen J. Trowbridge
Shinji Tsuji
Brad Turner
Kiyoshi Uematsu
Sterling A. Vaden
Magesh Valliappan
Schelto van Doorn
Anoop Vetteth
Chenxi Wang
Bill Woodruff
Hajime Yamashita
Yinglin (Frank) Yang
Bin Yeong Yoon
George Zimmerman

Historical participants

Included is a historical list of participants from the IEEE 802.3 and IEEE 802.1 Working Groups who have dedicated their valuable time, energy, and knowledge to the creation of IEEE Std 802.3ad-2000 Link Aggregation amendment from which this standard is derived.

Geoffrey O. Thompson, *IEEE 802.3 Working Group Chair*
David J. Law, *IEEE 802.3 Working Group Vice Chair*
Robert M. Grow, *IEEE 802.3 Working Group Secretary*
Steven R. Haddock, *Chair, IEEE P802.3ad Task Force*
Tony Jeffree, *Co-editor, IEEE P802.3ad Task Force*
Rich Seifert, *Co-editor, IEEE P802.3ad Task Force*

Oscar Agazzi
Don Alderrou
Abe Ali
Khaled Amer
Ralph Andersson
Ken-ichi Arai
Kameran Azadet
Keith Balmer
Denis Beaudoin
John L. Bestel
Michel Bohbot
Brad J. Booth
Paul Bottorff
Gary Bourque
Steve Brewer
Vince Bridgers
Benjamin Brown
Lisa Buckman
Robert Busse
Richard Cam
Bob Campbell
Edward G. Chang
Edward S. Chang
Hon Wan Chin
Chris Christ
Terry Cobb
Edward Cornejo
John Creigh
Kevin Daines
Tom Debiec
Chris Di Minico
Erik Dickens
Thomas J. Dineen
Dan Dove
Steve Dreyer
George Eisler
Michael Elswick
John F. Ewen
Mark Feuerstraeter
Jens Fiedler
Norival Figueira
Norman Finn
Alan Flatman
Howard M. Frazier
Scott Fritz
Darrell Furlong
Giorgio Giaretta
Joel Goergen

Bin Guo
Atikem Haile-Mariam
Sharam Hakimi
Del Hanson
Doug Harshbarger
Lloyd Hasley
Marwan Hassoun
Mehdi Hatamian
Gaby Hecht
Ken Herrity
John Hill
Henry Hinrichs
Clarence Joh
Thomas K. Jørgensen
Juan Jover
Paul Jury
Shinkyo Kaku
Mohan Kalkunte
Amrit Kalla
Hadriel Kaplan
Jaime Kardontchik
Toyoyuki Kato
Daniel Kelley
Yongbum Kim
Atsushi Kimoto
Keith Klamm
Christine Koenig
Paul F. Kolesar
Kishan Rao Konda
Josef Kosilek
Hans Lackner
Daun Langston
Loren Larsen
Brian E. Lemoff
Bill Lidinsky
George Lin
Jeffrey Lynch
Brian MacLeod
Kenneth MacLeod
Rabih Makarem
Robert A. Marsland
David W. Martin
Thomas Mathey
Tremont Miao
Colin Mick
Larry Miller
Cindy Montstream
Robert Mortonson
Simon Moseley
Shimon Muller
Denis Murphy
Yaron Nachman
Ken Naganuma
Hari Naidu
Paul Nikolich
Michael Nootbaar
Bob Noseworthy
Mark Nowell
Satoshi Obara
Toshio Ooka
Don Pannell
Luc Pariseau
John Payne
Robert Pieters
John Proffitt
Steve Pryor
William Quackenbush
Sailesh K. Rao
Peter Rautenberg
Anil Rijsinghani
Carlos Rodriguez
Shawn Rogers
Floyd E. Ross
Tam Ross
Larry Rubin
Peter Sallaway
Bill Sarles
J. David Schell
Ted Schroeder
Mick Seaman
Lee Sendelbach
Koichiro Seto
Som Sikdar
Alexander Smith
Andrew Smith
Michael A. Smith
David Sorensen
Michel Sørenson
Ronald Steudler
Stephen Strong
Steve Swanson
Tad Szostak
Rich Taborek
Pat Thaler
R. Jonathan Thatcher
Walter Thirion
Douglas Thomson
Bruce Tolley
Zbigniew Turlej
Edward Turner
Schelto van Doorn
Dono Van-Mierop
Nader Vijeh
Greg Wang
Peter Wang
Jim Welch
Willem Wery
Bob Williams
Joris Wils
John Wolcott
David Wong
Michael Wrigh
Chong Ho Yoon
Leonard Young
Ben Yu
Jing-fan Zhang

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

Thomas Alexander
Danilo Antonelli
Hugh Barrass
Michael Bennett
Rahul Bhushan
Tomo Bogataj
Ralf-Peter Braun
William Byrd
Jeffrey Cain
Edward Carley
James Carlo
Juan Carreon
Keith Chow
Alexander Conta
Charles Cook
John Dallesasse
John Dambrosia
Wael William Diab
Russell Dietz
Thomas Dineen
Marc Emmelmann
Jose Dominic Espejo
Prince Francis
Howard M. Frazier
Yukihiro Fujimoto
Devon Gayle
Michael Geipel
Sergiu Goma
Randall Groves
Robert M. Grow
C. Guy
Joseph Gwinn
John Hawkins
Adam Healey
Atsushi Ito
Raj Jain
Tony Jeffree
David Johnston
Peter Jones
Piotr Karocki
Stuart J. Kerry
Yongbum Kim
Paul Kolesar
Thomas Kurihara
David J. Law
Juan L. Lazaro
John Lemon
William Lumpkins
Mark Maloney
Peter Martini
W. Kyle Maus
Brett Mcclellan
Jonathon Mclendon
John Messenger
Jose Morales
Joseph Moran
Michael S. Newman
Nick S. A. Nikjoo
Satoshi Obara
George Oulundsen, III
Maximilian Riegel
Jessy V. Rouyer
Randall Safier
John Sauer
Benson Schliesser
Marco Scorrano
Michael Seaman
Thomas Starai
Marian Stasney
Walter Struppler
Alourdes Sully
Robert Sultan
Joseph Tardo
Michael Johas Teener
Geoffrey O. Thompson
Oliver Thorp
Thomas Tullia
Edward J. Turner
Mark-Rene Uchida
Schelto van Doorn
David Willow
Michael Wright
Takahito Yoshizawa
Oren Yuen
Paolo Zangheri

When the IEEE-SA Standards Board approved this standard on 26 September 2008, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Victor Berman
Richard DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
William R. Goldbach
Arnold M. Greenspan
Kenneth S. Hanus
Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Glenn Parsons
Ronald C. Petersen
Chuck Powers
Narayanan Ramachandran
Jon Walter Rosdahl
Robby Robson
Anne-Marie Sahazizia
Malcolm V. Thaden
Howard L. Wolfman
Don Wright

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Michael H. Kelly, *NIST Representative*

Michelle D. Turner
IEEE Standards Program Manager, Document Development

Michael D. Kipness
IEEE Standards Program Manager, Technical Program Development

List of special symbols

For the benefit of those who have received this document by electronic means, what follows is a list of special symbols and operators. If any of these symbols or operators fail to print out correctly on your machine, the editors apologize, and hope that this table will at least help you to sort out the meaning of the resulting funny-shaped blobs and strokes.

Special symbols and operators

Printed character	Meaning	Font
*	Boolean AND	Symbol
+	Boolean OR, arithmetic addition	Symbol
^	Boolean XOR	Times New Roman
!	Boolean NOT	Symbol
×	Multiplication	Symbol
<	Less than	Symbol
≤	Less than or equal to	Symbol
>	Greater than	Symbol
≥	Greater than or equal to	Symbol
=	Equal to	Symbol
≠	Not equal to	Symbol
⇐	Assignment operator	Symbol
∈	Indicates membership	Symbol
∉	Indicates nonmembership	Symbol
±	Plus or minus (a tolerance)	Symbol
°	Degrees	Symbol
∑	Summation	Symbol
√	Square root	Symbol
—	Big dash (em dash)	Times New Roman
–	Little dash (en dash), subtraction	Times New Roman
|	Vertical bar	Times New Roman
†	Dagger	Times New Roman
‡	Double dagger	Times New Roman
α	Lower case alpha	Symbol
β	Lower case beta	Symbol
γ	Lower case gamma	Symbol
δ	Lower case delta	Symbol
ε	Lower case epsilon	Symbol
λ	Lambda	Symbol
μ	Micro	Times New Roman
Ω	Omega	Symbol

Contents

IEEE Standard for
Local and metropolitan area networks—

Link Aggregation

1. Overview

1.1 Scope

Link Aggregation allows one or more links to be aggregated together to form a Link Aggregation Group, such that a MAC Client can treat the Link Aggregation Group as if it were a single link. To this end, it specifies the establishment of data terminal equipment (DTE) to DTE logical links, consisting of N parallel instances of full duplex point-to-point links operating at the same data rate. This standard defines the MAC independent Link Aggregation capability, and general information relevant to specific MAC types that support Link Aggregation.

1.2 Purpose

Link Aggregation allows the establishment of full duplex point-to-point links that have a higher aggregate bandwidth than the individual links that form the aggregation. This allows improved utilization of available links in bridged LAN environments, along with improved resilience in the face of failure of individual links

.

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 text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 802®-2001, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture.[1, 2]

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

IEEE Std 802.3™, IEEE Standard for 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.

IETF RFC 1155, *Structure and Identification of Management Information for TCP/IP-based Internets,* Rose, M., and K. McCloghrie, May 1990.[3]

IETF RFC 1157, *A Simple Network Management Protocol (SNMP),* Case, J., Fedor, M., Schoffstall, M., and J. Davin, May 1990.

IETF RFC 1212, *Concise MIB Definitions,* Rose, M., and K. McCloghrie, March 1991.

IETF RFC 1213 (IETF STD 17), *Management Information Base for Network Management of TCP/IP-based internets: MIB-II,* McCloghrie K., and M. Rose, Editors, March 1991.

IETF RFC 1215, *A Convention for Defining Traps for use with the SNMP,* M. Rose, March 1991.

IETF RFC 1901, *Introduction to Community-based SNMPv2,* Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, January 1996.

ISO/IEC 8824: 1990, Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1).[4]

ISO/IEC 10165-2: 1992, Information technology—Open Systems Interconnection—Structure of management information: Definition of management information.

ISO/IEC 10165-4: 1992, Information technology—Open Systems Interconnection—Management information services—Structure of management information—Part 4: Guidelines for the definition of managed objects.

ISO/IEC 15802-1: 1995, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 1: Medium Access Control (MAC) service definition.

---

[1]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ, 08854, USA (http://standards.ieee.org/).

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

[3]IETF RFCs are available from the Internet Engineering Task Force website at http://www.ietf.org/rfc.html.

[4]ISO/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 publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (http://www.ansi.org/).

3. Definitions

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standard Terms* (IEEE 100) should be referenced for terms not defined in this standard.

3.1 Actor: The local entity in a Link Aggregation Control Protocol exchange. (See IEEE Std 802.1AX, Clause 5.)

3.2 agent: A term used to refer to the managed nodes in a network. Managed nodes are those nodes that contain a network management entity (NME), which can be used to configure the node and/or collect data describing operation of that node. The agent is controlled by a network control host or manager that contains both an NME and network management application (NMA) software to control the operations of agents. Agents include systems that support user applications as well as nodes that provide communications services such as front-end processors, bridges, and routers. (See IEEE Std 802.3, Clause 30.[5])

3.3 Aggregation Key: A parameter associated with each port and with each Aggregator of an Aggregation System identifying those ports that can be aggregated together. Ports in an Aggregation System that share the same Aggregation Key value are potentially able to aggregate together. (See IEEE 802.1AX, Clause 5.)

3.4 Aggregation Link: An instance of a MAC-Physical Layer-Medium Physical Layer-MAC entity between a pair of Aggregation Systems. (See IEEE Std 802.1AX, Clause 5.)

3.5 Aggregation Port: An instance of a MAC-Physical Layer entity within an Aggregation System. (See IEEE Std 802.1AX, Clause 5.)

3.6 Aggregation System: A uniquely identifiable entity comprising (among other things) an arbitrary grouping of one or more ports for the purpose of aggregation. An instance of an aggregated link always occurs between exactly two Aggregation Systems. A physical device may comprise a single Aggregation System or more than one Aggregation System. (See IEEE Std 802.1AX, Clause 5.)

3.7 bridge: A layer 2 interconnection device that does not form part of a CSMA/CD collision domain but conforms to IEEE Std 802.1D. A bridge does not form part of a CSMA/CD collision domain but, rather appears as a Media Access Control (MAC) to the collision domain.

3.8 conversation: A set of frames transmitted from one end station to another, where all of the frames form an ordered sequence, and where the communicating end stations require the ordering to be maintained among the set of frames exchanged. (See IEEE Std 802.1AX, Clause 5.)

3.9 data terminal equipment (DTE): Any source or destination of data connected to the local area network.

3.10 end station: A system attached to a LAN that is an initial source or a final destination of frames transmitted across that LAN. A Network layer router is, from the perspective of the LAN, an end station; a bridge, in its role of forwarding frames from one LAN to another, is not an end station. (See IEEE Std 802.1AX, Clause 5.)

3.11 frame: Consists of the Destination Address, Source Address, Length/Type field, MAC Client Data, Pad (if required), and Frame Check Sequence.

3.12 full duplex: A mode of operation of a network, DTE, or Medium Attachment Unit (MAU) that supports duplex transmission as defined in IEEE 100. Within the scope of this standard, this mode of

[5]Information on references can be found in Clause 2.

operation allows for simultaneous communication between a pair of stations, provided that the Physical Layer is capable of supporting simultaneous transmission and reception without interference. (See IEEE Std 802.3.)

3.13 half duplex: A mode of operation of a CSMA/CD local area network (LAN) in which DTEs contend for access to a shared medium. Multiple, simultaneous transmissions in a half duplex mode CSMA/CD LAN result in interference, requiring resolution by the CSMA/CD access control protocol. (See IEEE Std 802.3.)

3.14 Key: *See:* **Aggregation Key.**

3.15 link: *See:* **Aggregation Link.**

3.16 Link Aggregation Group: A group of links that appear to a MAC Client as if they were a single link. All links in a Link Aggregation Group connect between the same pair of Aggregation Systems. One or more conversations may be associated with each link that is part of a Link Aggregation Group. (See IEEE Std 802.1AX, Clause 5.)

3.17 Management Information Base (MIB): A repository of information to describe the operation of a specific network device.

3.18 Media Access Control (MAC): The data link sublayer that is responsible for transferring data to and from the Physical Layer.

3.19 Partner: The remote entity in a Link Aggregation Control Protocol exchange. (See IEEE Std 802.1AX, Clause 5.)

3.20 port: *See:* **Aggregation Port.**

3.21 switch: A layer 2 interconnection device that conforms to ISO/IEC 10038 [ANSI/IEEE 802.1D]. *Syn:* **bridge.**

3.22 System: *See:* **Aggregation System.**

3.23 Type/Length/Value (TLV): A short, variable length encoding of an information element consisting of sequential type, length, and value fields where the type field identifies the type of information, the length field indicates the length of the information field in octets, and the value field contains the information, itself. The type value is locally defined and needs to be unique within the protocol defined in this standard.

4. Acronyms and abbreviations

This standard contains the following abbreviations:

ANSI	American National Standards Institute
ASN.1	abstract syntax notation one as defined in ISO/IEC 8824: 1990
CMIP	common management information protocol as defined in ISO/IEC 9596-1: 1991
DA	destination address
DTE	data terminal equipment
FCS	frame check sequence
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LACP	Link Aggregation Control Protocol
LACPDU	Link Aggregation Control Protocol Data Unit
LAG	Link Aggregation Group
LAG ID	Link Aggregation Group Identifier
LAN	local area network
LLC	logical link control
LLID	logical link identifier
MAC	medium access control
MAN	metropolitan area network
MIB	management information base
NTT	Need To Transmit
PDU	Protocol Data Unit
PHY	Physical Layer entity sublayer
PICS	protocol implementation conformance statement
SA	source address
TLV	Type/Length/Value
UCT	unconditional transition

5. Link Aggregation

5.1 Overview

This clause defines an optional Link Aggregation sublayer for use with CSMA/CD MACs. Link Aggregation allows one or more links to be aggregated together to form a Link Aggregation Group, such that a MAC Client can treat the Link Aggregation Group as if it were a single link. To this end, Link Aggregation specifies the establishment of DTE to DTE logical links, consisting of N parallel instances of full duplex point-to-point links operating at the same data rate.

5.1.1 State diagram conventions

The following state diagram conventions are used throughout this standard.

The operation of a protocol can be described by subdividing the protocol into a number of interrelated functions. The operation of the functions can be described by state diagrams. Each diagram represents the domain of a function and consists of a group of connected, mutually exclusive states. Only one state of a function is active at any given time (see Figure 5–1).

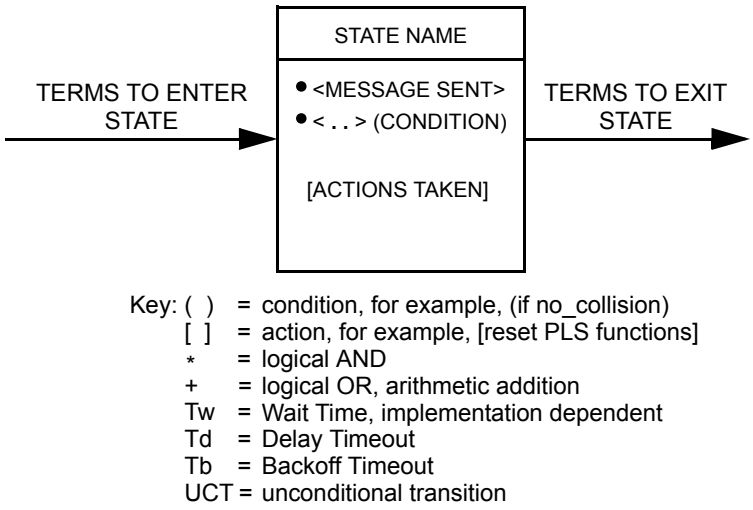


Figure 5–1—State diagram notation example

Each state that the function can assume is represented by a rectangle. These are divided into two parts by a horizontal line. In the upper part the state is identified by a name in capital letters. The lower part contains the name of any ON signal that is generated by the function. Actions are described by short phrases and enclosed in brackets.

All permissible transitions between the states of a function are represented graphically by arrows between them. A transition that is global in nature (for example, an exit condition from all states to the IDLE or RESET state) is indicated by an open arrow. Labels on transitions are qualifiers that must be fulfilled before the transition will be taken. The label UCT designates an unconditional transition. Qualifiers described by short phrases are enclosed in parentheses.

State transitions and sending and receiving of messages occur instantaneously. When a state is entered and the condition to leave that state is not immediately fulfilled, the state executes continuously, sending the messages in the state in a continuous manner.

The following conventions are used to describe a term-assignment statement that is associated with a transition:

a) The character ":" (colon) is a delimiter used to denote that a term assignment statement follows.
b) The character "⇐" (left arrow) denotes assignment of the value following the arrow to the term preceding the arrow.

The state diagrams contain the authoritative statement of the functions they depict; when apparent conflicts between descriptive text and state diagrams arise, the state diagrams are to take precedence. This does not override, however, any explicit description in the text that has no parallel in the state diagrams.

The models presented by state diagrams are intended as the primary specifications of the functions to be provided. It is important to distinguish, however, between a model and a real implementation. The models are optimized for simplicity and clarity of presentation, while any realistic implementation may place heavier emphasis on efficiency and suitability to a particular implementation technology. It is the functional behavior of any unit that must match the standard, not its internal structure. The internal details of the model are useful only to the extent that they specify the external behavior clearly and precisely.

5.1.1.1 Actions inside state blocks

The actions inside a state block execute instantaneously. Actions inside state blocks are atomic (i.e., uninterruptible).

After performing all the actions listed in a state block one time, the state block then continuously evaluates its exit conditions until one is satisfied, at which point control passes through a transition arrow to the next block. While the state awaits fulfillment of one of its exit conditions, the actions inside do not implicitly repeat.

The characters • and [bracket] are *not* used to denote any special meaning.

Valid state actions may include .indication and .request messages.

No actions are taken outside of any state block.

5.1.1.2 State diagram variables

Once set, variables retain their values as long as succeeding blocks contain no references to them.

Setting the parameter of a formal interface message assures that, on the next transmission of that message, the last parameter value set will be transmitted.

Testing the parameter of a formal interface messages tests the value of that message parameter that was received on the last transmission of said message. Message parameters may be assigned default values that persist until the first reception of the relevant message.

5.1.1.3 State transitions

The following terms are valid transition qualifiers:

a) Boolean expressions
b) An event such as the expiration of a timer: timer_done
c) An event such as the reception of a message: PMA_UNITDATA.indication
d) An unconditional transition: UCT
e) A branch taken when other exit conditions are not satisfied: ELSE

Any open arrow (an arrow with no source block) represents a global transition. Global transitions are evaluated continuously whenever any state is evaluating its exit conditions. When a global transition becomes true, it supersedes all other transitions, including UCT, returning control to the block pointed to by the open arrow.

5.1.1.4 Operators

The state machine operators are shown in Table 5–1.

Table 5–1—State machine operators

Character	Meaning
*	Boolean AND
+	Boolean OR
∧	Boolean XOR
!	Boolean NOT
<	Less than
≤	Less than or equal to
=	Equals (a test of equality)
≠	Not equals
≥	Greater than or equal to
>	Greater than
( )	Indicates precedence
⇐	Assignment operator
∈	Indicates membership
∉	Indicates nonmembership
|	Catenate
ELSE	No other state condition is satisfied

5.1.2 Goals and objectives

Link Aggregation, as specified in this clause, provides the following:

a) **Increased bandwidth**—The capacity of multiple links is combined into one logical link.

b) **Linearly incremental bandwidth**—Bandwidth can be increased in unit multiples as opposed to the order-of-magnitude increase available through Physical Layer technology options (10 Mb/s, 100 Mb/s, 1000 Mb/s, etc.).

c) **Increased availability**—The failure or replacement of a single link within a Link Aggregation Group need not cause failure from the perspective of a MAC Client.

d) **Load sharing**—MAC Client traffic may be distributed across multiple links.

e) **Automatic configuration**—In the absence of manual overrides, an appropriate set of Link Aggregation Groups is automatically configured, and individual links are allocated to those groups. If a set of links can aggregate, they will aggregate.

f) **Rapid configuration and reconfiguration**—In the event of changes in physical connectivity, Link Aggregation will quickly converge to a new configuration, typically on the order of 1 second or less.

g) **Deterministic behavior**—Depending on the selection algorithm chosen, the configuration can be made to resolve deterministically; i.e., the resulting aggregation can be made independent of the order in which events occur, and be completely determined by the capabilities of the individual links and their physical connectivity.

h) **Low risk of duplication or misordering**—During both steady-state operation and link (re)configuration, there is a low probability that frames are duplicated or misordered.

i) **Support of existing IEEE 802.3 MAC Clients**—No change is required to existing higher-layer protocols or applications to use Link Aggregation.

j) **Backwards compatibility with aggregation-unaware devices**—Links that cannot take part in Link Aggregation—either because of their inherent capabilities, management configuration, or the capabilities of the devices to which they attach—operate as normal, individual IEEE 802.3 links.

k) **Accommodation of differing capabilities and constraints**—Devices with differing hardware and software constraints on Link Aggregation are, to the extent possible, accommodated.

l) **No change to the IEEE 802.3 frame format**—Link aggregation neither adds to, nor changes the contents of frames exchanged between MAC Clients.

m) **Network management support**—The standard specifies appropriate management objects for configuration, monitoring, and control of Link Aggregation.

Link Aggregation, as specified in this clause, does not support the following:

n) **Multipoint Aggregations**—The mechanisms specified in this clause do not support aggregations among more than two Systems.

o) **Dissimilar MACs**—Link Aggregation is supported only on links using the IEEE 802.3 MAC.

p) **Half-duplex operation**—Link Aggregation is supported only on point-to-point links with MACs operating in full duplex mode.

q) **Operation across multiple data rates**—All links in a Link Aggregation Group operate at the same data rate (e.g., 10 Mb/s, 100 Mb/s, or 1000 Mb/s).

5.1.3 Positioning of Link Aggregation within the IEEE 802.3 architecture

Link Aggregation comprises an optional sublayer between a MAC Client and the MAC (or optional MAC Control sublayer). Figure 5–2 depicts the positioning of the Link Aggregation sublayer in the CSMA/CD layer architecture, and the relationship of that architecture to the Data Link and Physical Layers of the OSI Reference Model. The figure also shows the ability of the Link Aggregation sublayer to combine a number of individual links in order to present a single MAC interface to the MAC Client.

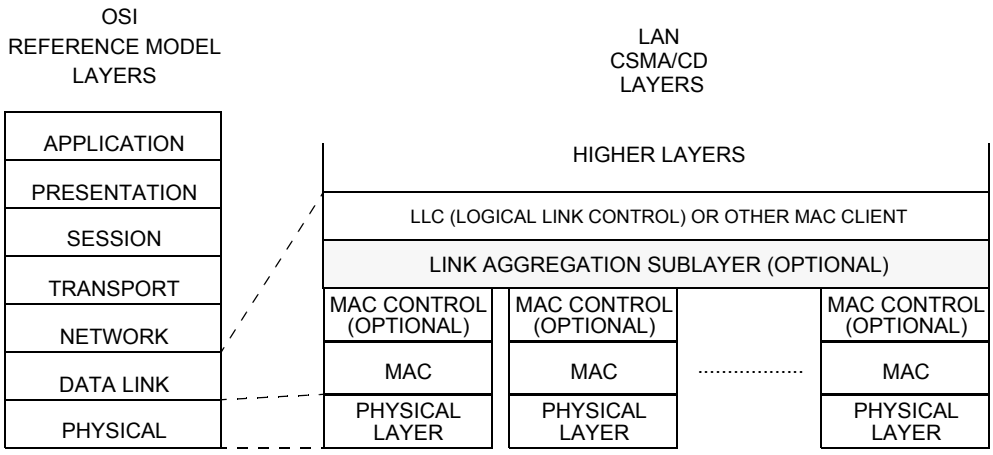


Figure 5–2—Architectural positioning of Link Aggregation sublayer

Figure 5–3 depicts the major blocks that form the Link Aggregation sublayer, and their interrelationships.

It is possible to implement the optional Link Aggregation sublayer for some ports within a System while not implementing it for other ports; i.e., it is not necessary for all ports in a System to be subject to Link Aggregation. A conformant implementation is not required to be able to apply the Link Aggregation sublayer to every port.

5.2 Link Aggregation operation

As depicted in Figure 5–3, the Link Aggregation sublayer comprises the following functions:

a) *Frame Distribution.* This block is responsible for taking frames submitted by the MAC Client and submitting them for transmission on the appropriate port, based on a frame distribution algorithm employed by the Frame Distributor. Frame Distribution also includes an optional *Marker Generator/Receiver* used for the Marker protocol. (See 5.2.4, 5.2.5, and 5.5.)

b) *Frame Collection.* This block is responsible for passing frames received from the various ports to the MAC Client. Frame Collection also includes a *Marker Responder,* used for the Marker protocol. (See 5.2.3 and 5.5.)

c) *Aggregator Parser/Multiplexers.* On transmit, these blocks simply pass frame transmission requests from the Distributor, Marker Generator, and/or Marker Responder to the appropriate port. On receive, these blocks distinguish among Marker Request, Marker Response, and MAC Client PDUs, and pass each to the appropriate entity (Marker Responder, Marker Receiver, and Collector, respectively).

d) *Aggregator.* The combination of Frame Distribution and Collection, along with the Aggregator Parser/Multiplexers, is referred to as the Aggregator.

e) *Aggregation Control.* This block is responsible for the configuration and control of Link Aggregation. It incorporates a *Link Aggregation Control Protocol* (LACP) that can be used for automatic communication of aggregation capabilities between Systems and automatic configuration of Link Aggregation.

f) *Control Parser/Multiplexers.* On transmit, these blocks simply pass frame transmission requests from the Aggregator and Control entities to the appropriate port. On receive, these blocks distinguish Link Aggregation Control PDUs from other frames, passing the LACPDUs to the appropriate sublayer entity, and all other frames to the Aggregator.

5.2.1 Principles of Link Aggregation

Link Aggregation allows a MAC Client to treat a set of one or more ports as if it were a single port. In doing so, it employs the following principles and concepts:

a) A MAC Client communicates with a set of ports through an Aggregator, which presents a standard IEEE 802.3 service interface to the MAC Client. The Aggregator binds to one or more ports within a System.

b) It is the responsibility of the Aggregator to distribute frame transmissions from the MAC Client to the various ports, and to collect received frames from the ports and pass them to the MAC Client transparently.

c) A System may contain multiple Aggregators, serving multiple MAC Clients. A given port will bind to (at most) a single Aggregator at any time. A MAC Client is served by a single Aggregator at a time.

d) The binding of ports to Aggregators within a System is managed by the Link Aggregation Control function for that System, which is responsible for determining which links may be aggregated, aggregating them, binding the ports within the System to an appropriate Aggregator, and monitoring conditions to determine when a change in aggregation is needed.

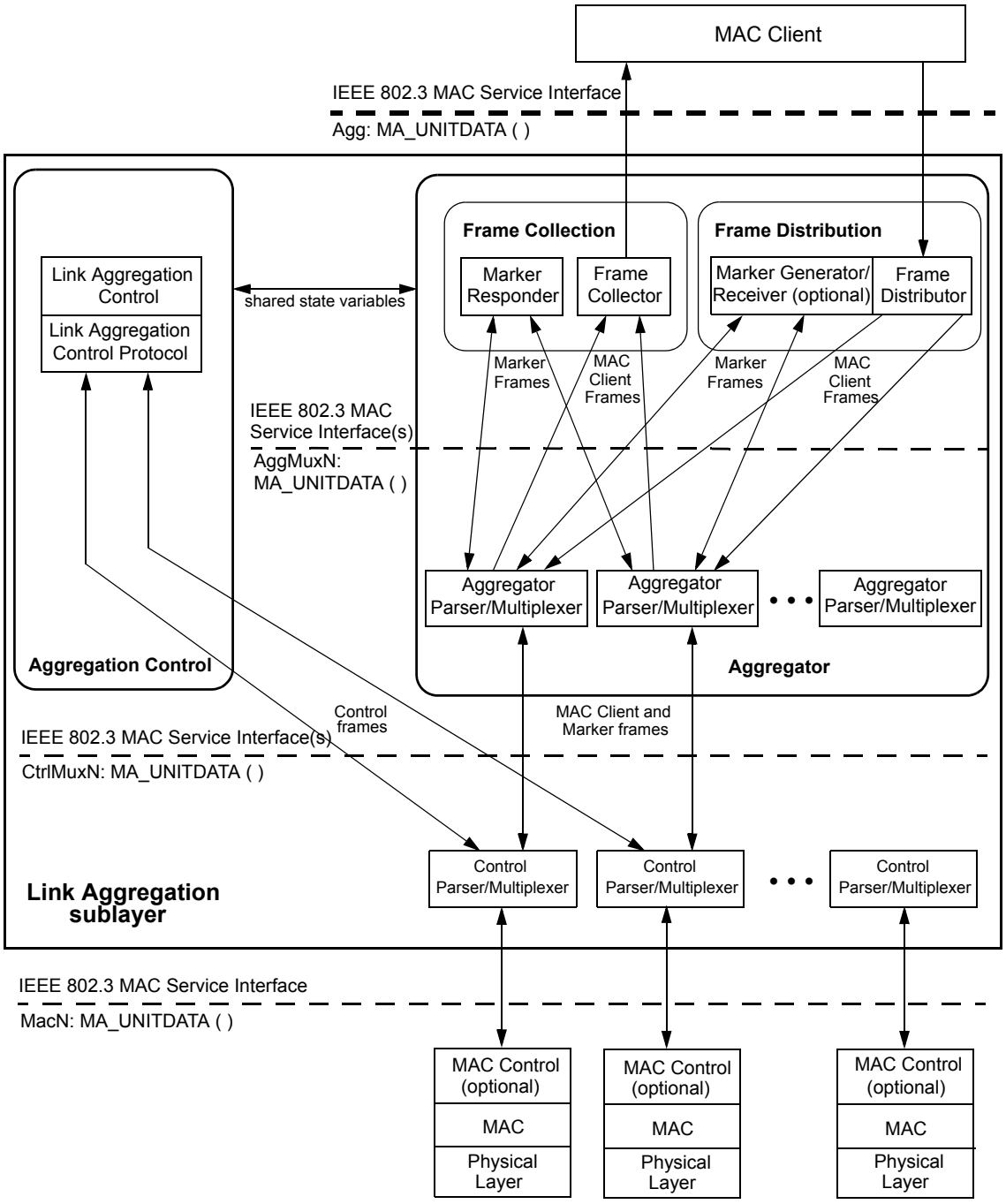


Figure 5–3—Link Aggregation sublayer block diagram

e) Such determination and binding may be under manual control through direct manipulation of the state variables of Link Aggregation (e.g., Keys) by a network manager. In addition, automatic determination, configuration, binding, and monitoring may occur through the use of a Link Aggregation Control Protocol (LACP). The LACP uses peer exchanges across the links to determine, on an ongoing basis, the aggregation capability of the various links, and continuously provides the maximum level of aggregation capability achievable between a given pair of Systems.

f) Frame ordering must be maintained for certain sequences of frame exchanges between MAC Clients (known as conversations, see Clause 3). The Distributor ensures that all frames of a given conversation are passed to a single port. For any given port, the Collector is required to pass frames to the MAC Client in the order that they are received from that port. The Collector is otherwise free to select frames received from the aggregated ports in any order. Since there are no means for frames to be misordered on a single link, this guarantees that frame ordering is maintained for any conversation.

g) Conversations may be moved among ports within an aggregation, both for load balancing and to maintain availability in the event of link failures.

h) This standard does not impose any particular distribution algorithm on the Distributor. Whatever algorithm is used should be appropriate for the MAC Client being supported.

i) Each port is assigned a unique, globally administered MAC address. This MAC address is used as the source address for frame exchanges that are initiated by entities within the Link Aggregation sublayer itself (i.e., LACP and Marker protocol exchanges).

NOTE—The LACP and Marker protocols use a multicast destination address for all exchanges, and do not impose any requirement for a port to recognize more than one unicast address on received frames.[6]

j) Each Aggregator is assigned a unique, globally administered MAC address; this address is used as the MAC address of the aggregation from the perspective of the MAC Client, both as a source address for transmitted frames and as the destination address for received frames. The MAC address of the Aggregator may be one of the MAC addresses of a port in the associated Link Aggregation Group (see 5.2.10).

5.2.2 Service interfaces

The MAC Client communicates with the Aggregator using the standard service interface specified in IEEE Std 802.3 Clause 2. Similarly, Link Aggregation communicates internally (between Frame Collection/ Distribution, the Aggregator Parser/Multiplexers, the Control Parser/Multiplexers, and Link Aggregation Control) and with its bound ports using the same, standard service interface. No new interlayer service interfaces are defined for Link Aggregation.

Since Link Aggregation uses four instances of the MAC Service Interface, it is necessary to introduce a notation convention so that the reader can be clear as to which interface is being referred to at any given time. A prefix is therefore assigned to each service primitive, indicating which of the four interfaces is being invoked, as depicted in Figure 5–3. The prefixes are as follows:

a) *Agg:,* for primitives issued on the interface between the MAC Client and the Link Aggregation sublayer.

b) *AggMuxN:,* for primitives issued on the interface between Aggregator Parser/Multiplexer N and its internal clients (where N is the port number associated with the Aggregator Parser/Multiplexer).

c) *CtrlMuxN:,* for primitives issued on the interface between Control Parser/Multiplexer N and its internal clients (where N is the port number associated with the Control Parser/Multiplexer).

d) *MacN:,* for primitives issued on the interface between underlying MAC N and its Control Parser/ Multiplexer (where N is the port number associated with the underlying MAC).

MAC Clients may generate Agg:MA_DATA.request primitives for transmission on an aggregated link. These are passed by the Frame Distributor to a port selected by the distribution algorithm. MacN:MA_DATA.indication primitives signifying received frames are passed unchanged from a port to the MAC Client by the Frame Collector.

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

MAC Clients that generate MA_CONTROL.request primitives (and which expect MA_CONTROL.indication primitives in response) cannot communicate through a Link Aggregation sublayer. They must communicate directly with the MAC Control entity through which these control primitives are to be sent and received. The multiplexing of such MAC Control clients with a Link Aggregation sublayer for simultaneous use of a single port is outside the scope of this standard.

The multiplexing of MAC Clients with a Link Aggregation sublayer for simultaneous use of an individual MAC that is also part of a Link Aggregation Group is outside the scope of this standard.

5.2.3 Frame Collector

A Frame Collector is responsible for receiving incoming frames (i.e., AggMuxN:MA_DATA.indications) from the set of individual links that form the Link Aggregation Group (through each link's associated Aggregator Parser/Multiplexer) and delivering them to the MAC Client. Frames received from a given port are delivered to the MAC Client in the order that they are received by the Frame Collector. Since the Frame Distributor is responsible for maintaining any frame ordering constraints, there is no requirement for the Frame Collector to perform any reordering of frames received from multiple links.

The Frame Collector shall implement the function specified in the state diagram shown in Figure 5–4 and the associated definitions contained in 5.2.3.1.

5.2.3.1 Frame Collector state diagram

5.2.3.1.1 Constants

CollectorMaxDelay

In tens of microseconds, the maximum time that the Frame Collector may delay the delivery of a frame received from an Aggregator Parser to its MAC Client. Value is assigned by management or administration policy.

Value: Integer

5.2.3.1.2 Variables

DA

SA

mac_service_data_unit

status

The parameters of the MA_DATA.indication primitive, as defined in IEEE Std 802.3 Clause 2.

BEGIN

A Boolean variable that is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.

Value: Boolean

5.2.3.1.3 Messages

Agg:MA_DATA.indication
AggMuxN:MA_DATA.indication

The service primitives used to pass a received frame to a client with the specified parameters.

5.2.3.1.4 State diagram

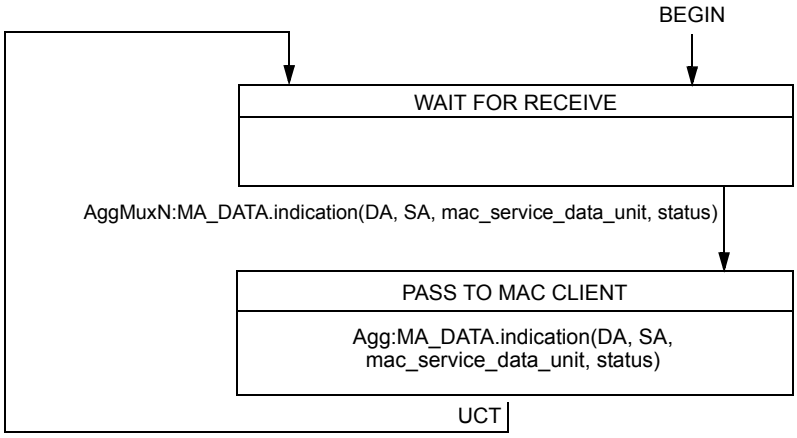


Figure 5–4—Frame Collector state diagram

The architectural models of the IEEE 802.3 MAC and Link Aggregation do not make any provision for queueing of frames between the link and the MAC Client. Furthermore, the state machine conventions used in this standard assume that actions within a state occur instantaneously (see 5.1.1). However, practical implementations of Link Aggregation will typically incur both queueing and delay in the Frame Collector. In order to ensure that frame delivery is not delayed indefinitely (which could cause a frame ordering problem when moving conversations from one link to another), the Frame Collector shall, upon receiving a frame from an Aggregator Parser, either deliver the frame to its MAC Client, or discard the frame within a CollectorMaxDelay time. The Frame Distributor (within the Partner System at the other end of the link) can assume that all frames transmitted on a given link have been either received by its Partner's MAC Client or discarded after a CollectorMaxDelay plus the propagation delay of the link. The use of CollectorMaxDelay is further discussed in A.3.

5.2.4 Frame Distributor

The Frame Distributor is responsible for taking outgoing frames from the MAC Client and transmitting them through the set of links that form the Link Aggregation Group. The Frame Distributor implements a distribution function (algorithm) responsible for choosing the link to be used for the transmission of any given frame or set of frames.

This standard does not mandate any particular distribution algorithm(s); however, any distribution algorithm shall ensure that, when frames are received by a Frame Collector as specified in 5.2.3, the algorithm shall not cause

a) Misordering of frames that are part of any given conversation, or
b) Duplication of frames.

The above requirement to maintain frame ordering is met by ensuring that all frames that compose a given conversation are transmitted on a single link in the order that they are generated by the MAC Client; hence, this requirement does not involve the addition (or modification) of any information to the MAC frame, nor any buffering or processing on the part of the corresponding Frame Collector in order to reorder frames. This approach to the operation of the distribution function permits a wide variety of distribution and load balancing algorithms to be used, while also ensuring interoperability between devices that adopt differing algorithms.

NOTE—The subject of distribution algorithms and maintenance of frame ordering is discussed in Annex A.

The Frame Distributor shall implement the function specified in the state diagram shown in Figure 5–5 and the associated definitions contained in 5.2.4.1.

5.2.4.1 Frame Distributor state diagram

5.2.4.1.1 Variables

DA

SA

mac_service_data_unit

The parameters of the MA_DATA.request primitive, as defined in IEEE Std 802.3 Clause 2.

BEGIN

A Boolean variable that is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.

Value: Boolean

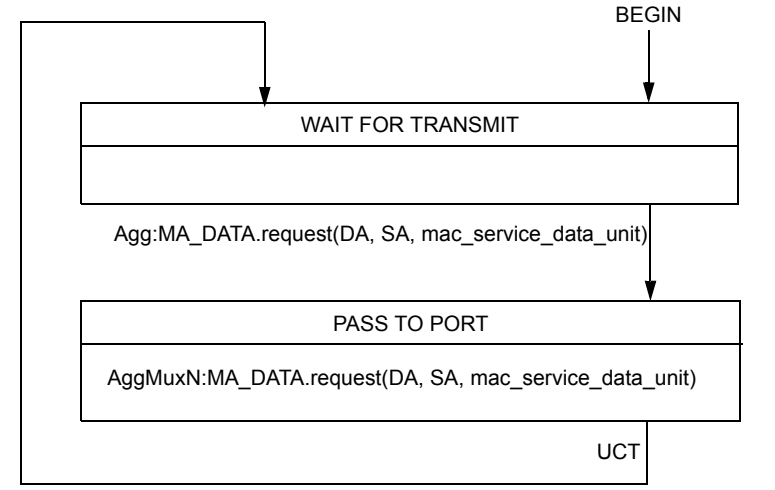
5.2.4.1.2 Messages

Agg:MA_DATA.request

AggMuxN:MA_DATA.request

The service primitives used to transmit a frame with the specified parameters.

5.2.4.1.3 State diagram



If a client issues an Agg:MA_DATA.request primitive that contains no SA parameter, the AggMuxN:MA_DATA.request primitive generated shall use the Aggregator's MAC address for the SA.

NOTE—The algorithm that the Frame Distributor uses to select the value of N in AggMuxN:MA_DATA.request for a given frame is unspecified.

Figure 5–5—Frame Distributor state diagram

5.2.5 Marker Generator/Receiver (optional)

The optional Marker Generator is used by the Marker protocol, as specified in 5.5. When implemented and so requested by the Distribution algorithm, the Marker Generator shall issue an AggMuxN:MA_DATA.request primitive, with a mac_service_data_unit containing a Marker PDU as defined in 5.5.3, to the port associated with the conversation being marked, subject to the timing restrictions for Slow Protocols specified in IEEE Std 802.3 Annex 57A.

The optional Marker Receiver is used by the Marker protocol, as specified in 5.5. It receives Marker Response PDUs from the Aggregator Parser.

5.2.6 Marker Responder

The Marker Responder is used by the Marker protocol, as specified in 5.5. The Marker Responder receives Marker PDUs (generated by a Partner System's Marker Generator), and transmits a Marker Response PDU through the same port from which the Marker PDU was received. While implementation of the Marker Generator/Receiver is optional, the ability to respond to a Marker PDU (the Marker Responder) is mandatory. An implementation conformant to this clause shall implement the Marker Responder as specified in 5.5.4.2, thus ensuring that implementations that need to make use of the protocol can do so.

5.2.7 Aggregator Parser/Multiplexer

On transmission, the Aggregator Multiplexer shall provide transparent pass-through of frames submitted by the Marker Responder and optional Marker Generator to the port specified in the transmission request. The Aggregator Multiplexer shall provide transparent pass-through of frames submitted by the Frame Distributor to the port specified in the transmission request only when the port state is Distributing (see 5.4.15); otherwise, such frames shall be discarded.

On receipt, the Aggregator Parser decodes frames received from the Control Parser, passes those frames destined for the Marker Responder or Marker Receiver to the selected entity, and discards frames with invalid Slow Protocol subtype values (see IEEE Std 802.3 Table 57A–2). The Aggregator Parser shall pass all other frames to the Frame Collector for passage to the MAC Client only when the port state is Collecting (see 5.4.15); otherwise, such frames shall be discarded. The Aggregator Parser shall implement the function specified in the state diagram shown in Figure 5–6 and the associated definitions contained in 5.2.7.1.

5.2.7.1 Aggregator Parser state diagram

5.2.7.1.1 Constants

Slow_Protocols_Multicast

The value of the Slow Protocols Multicast address. (See IEEE Std 802.3 Table 57A–1.)

Slow_Protocols_Type

The value of the Slow Protocols Length/Type field. (See IEEE Std 802.3 Annex 57A.)

Marker_subtype

The value of the Subtype field for the Marker protocol. (See 5.5.3.)

Value: Integer

2

Marker_Information

The encoding of the Marker Information TLV_type field. (See 5.5.3.)

Value: Integer

1

Marker_Response_Information

The encoding of the Marker Response Information TLV_type field. (See 5.5.3.)

Value: Integer

2

5.2.7.1.2 Variables

DA

SA

mac_service_data_unit

status

The parameters of the MA_DATA.indication primitive as defined in IEEE Std 802.3 Clause 2.

Length/Type

The value of the Length/Type field in a received frame.

Value: Integer

Subtype

The value of the octet following the Length/Type field in a Slow Protocol frame. (See IEEE Std 802.3 Annex 57A.)

Value: Integer

TLV_type

The value contained in the octet following the Version Number in a received Marker or Marker Response frame. This identifies the "type" for the Type/Length/Value (TLV) tuple. (See 5.5.3.)

Value: Integer

BEGIN

A Boolean variable that is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.

Value: Boolean

5.2.7.1.3 Messages

CtrlMuxN:MA_DATA.indication

AggMuxN:MA_DATA.indication

The service primitives used to pass a received frame to a client with the specified parameters.

5.2.8 Aggregator

An *Aggregator* comprises an instance of a Frame Collection function, an instance of a Frame Distribution function and one or more instances of the Aggregator Parser/Multiplexer function for a Link Aggregation Group. A single Aggregator is associated with each Link Aggregation Group. An Aggregator offers a standard IEEE 802.3 MAC service interface to its associated MAC Client; access to the MAC service by a MAC Client is always achieved via an Aggregator. An Aggregator can therefore be considered to be a

logical MAC, bound to one or more ports, through which the MAC client is provided access to the MAC service.

A single, individual MAC address is associated with each Aggregator (see 5.2.10).

An Aggregator is available for use by the MAC Client if the following are all true:

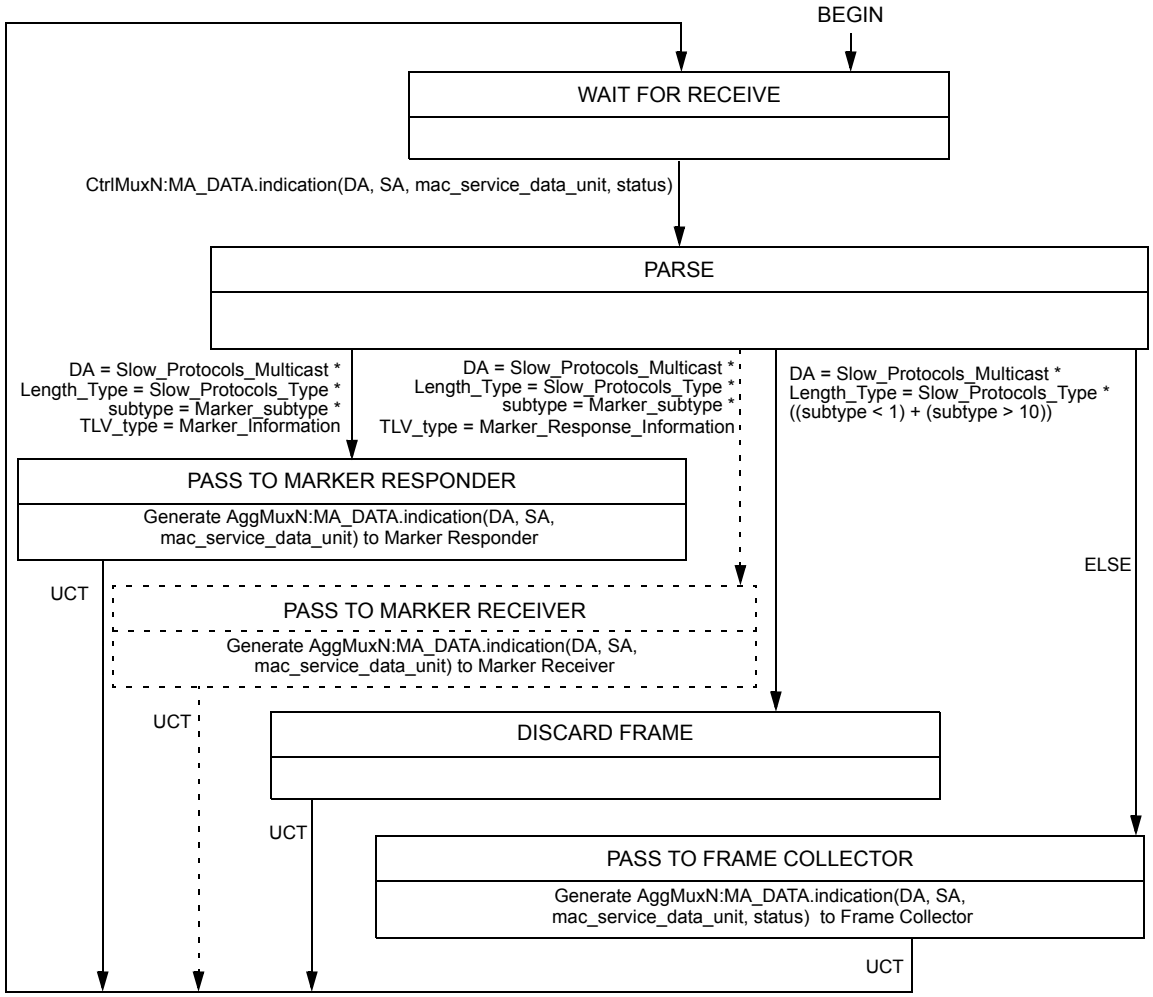
a) It has one or more attached ports.

b) The Aggregator has not been set to a disabled state by administrative action (see 6.3.1.1.13).

c) The collection and/or distribution function associated with one or more of the attached ports is enabled (see 6.3.1.1.14).

NOTE—To simplify the modeling and description of the operation of Link Aggregation, it is assumed that there are as many Aggregators as there are ports in a given System; however, this is not a requirement of this standard. Aggregation of two or more ports consists of changing the bindings between ports and Aggregators such that more than one port is bound to a single Aggregator. The creation of any aggregations of two or more links will therefore result in one or more Aggregators that are bound to more than one port, and one or more Aggregators that are not bound to any port. An Aggregator that is not bound to any port appears to a MAC Client as a MAC interface to an inactive port. During times when the bindings between ports and Aggregators are changing, or as a consequence of particular configuration choices, there may be occasions when one or more ports are not bound to any Aggregator.

5.2.8.1 State diagram



If the optional Marker Receiver is not implemented, Marker Responses shall be passed to the Frame Collector. If the port state is not Collecting, all frames that would have been passed to the MAC Client through the Collector will be discarded.

Figure 5–6—Aggregator Parser state diagram

5.2.9 Control Parser/Multiplexer

On transmission, the Control Multiplexer shall provide transparent pass-through of frames submitted by the Aggregator and Link Aggregation Control Protocol to the port specified in the transmission request.

On receipt, the Control Parser decodes frames received from the various ports in the Link Aggregation Group, passes those frames destined for the Link Aggregation Control Protocol to the appropriate entity, and passes all other frames to the Aggregator Parser. The Control Parser shall implement the function specified by the state diagram shown in Figure 5–7 and the associated definitions contained in 5.2.9.1.

5.2.9.1 Control Parser state diagram

5.2.9.1.1 Constants

Slow_Protocols_Multicast

The value of the Slow Protocols Multicast address. (See IEEE Std 802.3 Table 57A–1.)

Slow_Protocols_Type

The value of the Slow Protocols Length/Type field. (See IEEE Std 802.3 Table 57A–2.)

LACP_subtype

The value of the Subtype field for the Link Aggregation Control Protocol. (See IEEE Std 802.3 Table 57A–3.)

Value: Integer

1

5.2.9.1.2 Variables

DA

SA

mac_service_data_unit

status

The parameters of the MA_DATA.indication primitive, as defined in IEEE Std 802.3 Clause 2.

Length/Type

The value of the Length/Type field in a received frame.

Value: Integer

Subtype

The value of the octet following the Length/Type field in a Slow Protocol frame.
(See IEEE Std 802.3 Annex 57A.)

Value: Integer

BEGIN

A Boolean variable that is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.

Value: Boolean

5.2.9.1.3 Messages

MacN:MA_DATA.indication

CtrlMuxN:MA_DATA.indication

The service primitives used to pass a received frame to a client with the specified parameters.

5.2.9.1.4 State diagram

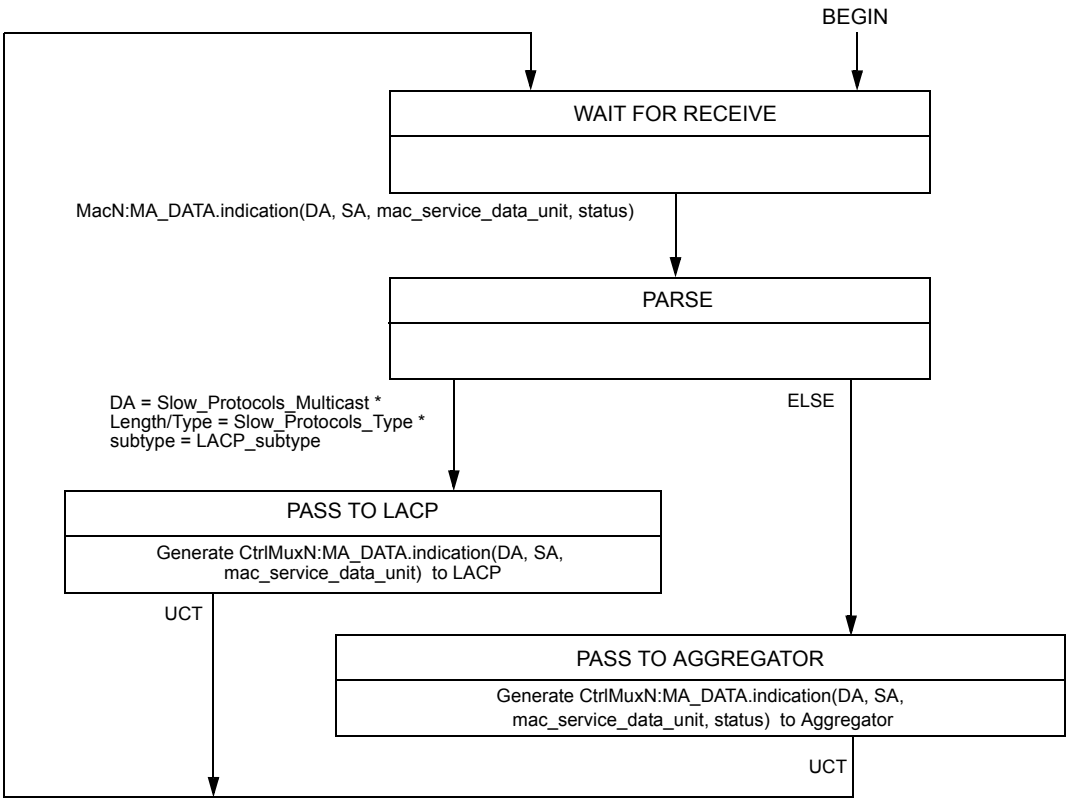


Figure 5–7—Control Parser state diagram

5.2.10 Addressing

Each IEEE 802.3 MAC has an associated individual MAC address, whether that MAC is used for Link Aggregation or not (see IEEE Std 802.3 Clause 4).

Each Aggregator to which one or more ports are attached has an associated globally-unique individual MAC address (see 5.3.3). The MAC address of the Aggregator may be the globally-unique individual MAC addresses of one of the MACs in the associated Link Aggregation Group, or it may be a distinct MAC address. The manner in which such addresses are chosen is not otherwise constrained by this standard.

Protocol entities sourcing frames from within the Link Aggregation sublayer (e.g., LACP and the Marker protocol) use the MAC address of the MAC within an underlying port as the source address in frames transmitted through that port. The MAC Client sees only the Aggregator and not the underlying MACs, and therefore uses the Aggregator's MAC address as the source address in transmitted frames. If a MAC Client

submits a frame to the Aggregator for transmission without specifying a source address, the Aggregator inserts its own MAC address as the source address for transmitted frames.

NOTE—This behavior causes the Aggregator to behave the same way as a standard MAC with regard to frames submitted by its client.

5.3 Link Aggregation Control

Link Aggregation Control configures and controls the Link Aggregation sublayer using static information local to the control function and dynamic information exchanged by means of the Link Aggregation Control Protocol.

For each aggregateable port in the System, Link Aggregation Control

a) Maintains configuration information (reflecting the inherent properties of the individual links as well as those established by management) to control aggregation.

b) Exchanges configuration information with other Systems to allocate the link to a Link Aggregation Group.

NOTE—A given link is allocated to, at most, one Link Aggregation Group at a time. The allocation mechanism attempts to maximize aggregation, subject to management controls.

c) Attaches the port to the Aggregator used by the Link Aggregation Group, and detaches the port from the Aggregator when it is no longer used by the Group.

d) Uses information from the Partner System's Link Aggregation Control entity to enable or disable the Aggregator's Collector and Distributor.

The operation of Link Aggregation Control involves the following activities, which are described in detail in 5.3.1 through 5.3.15:

e) Checking that candidate links can actually be aggregated.

f) Controlling the addition of a link to a Link Aggregation Group, and the creation of the group if necessary.

g) Monitoring the status of aggregated links to ensure that the aggregation is still valid.

h) Removing a link from a Link Aggregation Group if its membership is no longer valid, and removing the group if it no longer has any member links.

In order to allow Link Aggregation Control to determine whether a set of links connect to the same System, and to determine whether those links are compatible from the point of view of aggregation, it is necessary to be able to establish

i) A globally unique identifier for each System that participates in Link Aggregation (see 5.3.2).

j) A means of identifying the set of capabilities associated with each port and with each Aggregator, as understood by a given System.

k) A means of identifying a Link Aggregation Group and its associated Aggregator.

System identification allows the detection of links that are connected in a loopback configuration (i.e., both ends of the same link are connected to the same System).

5.3.1 Characteristics of Link Aggregation Control

Link Aggregation Control provides a configuration capability that is

a) **Automatic.** In the absence of manual override controls, an appropriate set of Link Aggregation Groups is automatically configured, and individual links are allocated to those groups. If a set of links can aggregate, they do aggregate.

b) **Continuous.** Manual intervention or initialization events are not a requirement for correct operation. The configuration mechanism continuously monitors for changes in state that require reconfiguration. The configuration functions detect and correct misconfigurations by performing reconfiguration and/or by taking misconfigured links out of service.

c) **Deterministic.** The configuration can resolve deterministically; i.e., the configuration achieved can be made independent of the order in which events occur, and be completely determined by the combination of the capabilities of the individual links and their physical connectivity.

d) **Controllable.** The configuration capabilities accommodate devices with differing hardware and software constraints on Link Aggregation.

e) **Compatible.** Links that cannot take part in Link Aggregation, either because of their inherent capabilities or of the capabilities of the devices to which they attach, operate as normal IEEE 802.3 links. The introduction of Link Aggregation capability at one or both ends of a link should not result in a degradation of the perceived performance of the link.

f) **Rapid.** The configuration resolves rapidly to a stable configuration. Convergence can be achieved by the exchange of three LACPDUs, without dependence on timer values.

with

g) **Low risk of misdelivery.** The operation of the (re-)configuration functions minimizes the risk of frames being delivered to the wrong Aggregator.

h) **Low risk of duplication or misordering.** The operation of the (re-)configuration functions minimizes the risk of frame duplication and frame misordering.

i) **Low protocol overhead.** The overhead involved in external communication of configuration information between devices is small.

5.3.2 System identification

The globally unique identifier used to identify a System shall be the concatenation of a globally administered individual MAC address and the System Priority. The MAC address chosen may be the individual MAC address associated with one of the ports of the System.

Where it is necessary to perform numerical comparisons between System Identifiers, each System Identifier is considered to be an eight octet unsigned binary number, constructed as follows:

a) The two most significant octets of the System Identifier comprise the System Priority. The System Priority value is taken to be an unsigned binary number; the most significant octet of the System Priority forms the most significant octet of the System Identifier.

b) The third most significant octet of the System Identifier is derived from the initial octet of the MAC address; the least significant bit of the octet is assigned the value of the first bit of the MAC address, the next most significant bit of the octet is assigned the value of the next bit of the MAC address, and so on. The fourth through eighth octets are similarly assigned the second through sixth octets of the MAC address.

5.3.3 Aggregator identification

Each Aggregator to which one or more ports are attached shall be assigned a unique, globally administered individual MAC address. The MAC address assigned to the Aggregator may be the same as the MAC address assigned to one of its bound ports. No Aggregator shall be assigned a MAC address that is the same as that of a port bound to a different Aggregator within the System. When receiving frames, a port is never required to recognize more than one unicast address, i.e., the Aggregator's MAC address.

NOTE—This ensures that Aggregators can be uniquely addressed, and allows (but does not require) the unique address to be allocated from the same set of addresses as are assigned to the ports. It also acknowledges the fact that locally administered addresses may be used in particular implementations or environments. The stated restriction on the allocation of MAC addresses to Aggregators may have implications with regard to the choice of selection algorithm.

An Aggregator also shall be assigned an integer identifier that is used by Link Aggregation Control to uniquely identify the Aggregator within the System. This value will typically be the same as the interface identifier (ifIndex) used for management purposes.

5.3.4 Port identification

Link Aggregation Control uses a Port Identifier, comprising the concatenation of a Port Priority and a Port Number, to identify the port. Port Numbers (and hence, Port Identifiers) shall be uniquely assigned within a System. Port Number 0 shall not be assigned to any port.

When it is necessary to perform numerical comparisons between Port Identifiers, each Port Identifier is considered to be a four octet unsigned binary number constructed as follows:

a) The most significant and second most significant octets are the first and second most significant octets of the Port Priority, respectively.

b) The third and fourth most significant octets are the first and second most significant octets of the Port Number, respectively.

5.3.5 Capability identification

The ability of one port to aggregate with another is summarized by a simple integer parameter, known as a Key. This facilitates communication and comparison of aggregation capabilities, which may be determined by a number of factors, including

a) The port's physical characteristics, such as data rate, duplexity, and point-to-point or shared medium.

b) Configuration constraints established by the network administrator.

c) Use of the port by higher layer protocols (e.g., assignment of Network Layer addresses).

d) Characteristics or limitations of the port implementation itself.

Two Keys shall be associated with each port—an operational Key and an administrative Key. The operational Key is the Key that is currently in active use for the purposes of forming aggregations. The administrative Key allows manipulation of Key values by management. The administrative and operational Keys assigned to a port may differ

e) If the operation of the implementation is such that an administrative change to a Key value cannot be immediately reflected in the operational state of the port.

f) If the System supports the dynamic manipulation of Keys, as discussed in 5.6.2, either to accurately reflect changes in operational capabilities of the port (for example, as a result of Auto-Negotiation), or to provide a means of handling constraints on aggregation capability.

A given Key value is meaningful only in the context of the System that allocates it; there is no global significance to Key values. Similarly, the relationship between administrative and operational Key values is meaningful only in the context of the System that allocates it. When a System assigns an administrative Key value to a set of ports, it signifies that the set of ports have the potential to aggregate together, subject to the considerations discussed in 5.6.2. When a System assigns an operational Key value to a set of ports, it signifies that, in the absence of other constraints, the current operational state of the set of ports allows any subset of that set of ports (including the entire set) to be aggregated together from the perspective of the System making the assignment. The set of such ports that will actually be aggregated will be those that terminate at a common Partner System, and for which that Partner System has assigned a common operational Key value, local to that Partner. The set of ports in a given System that share the same operational Key value are said to be members of the same Key Group.

A System may determine that a given link is not able to be aggregated with other links. Such links are referred to as Individual links (as opposed to Aggregateable links). A System may declare a link to be

Individual if the inherent properties of the link allow its use as part of an aggregation, but the system is aware of no other links that are capable of aggregating with this link (e.g., the System has allocated a unique operational Key value to the link).

The capability information communicated between Systems, therefore, includes this local knowledge of the aggregation capability of the link in addition to the operational Key value; i.e., whether the System considers the link to be Aggregateable or Individual.

An administrative Key value and an operational Key value shall also be associated with each Aggregator. The operational Key is the Key that is currently in active use for the purposes of forming aggregations. The administrative Key allows manipulation of Key values by management. The values of administrative and operational Key for an Aggregator may differ in the same manner as that of port Keys, per item e) and item f) in this subclause. Ports that are members of a given Key Group can only be bound to Aggregators that share the same operational Key value.

All Keys are 16-bit identifiers. All values except the null value (all zeros) are available for local use.

NOTE—This model allows for two convenient initial configurations. The first is achieved by assigning each port an initial administrative and operational Key value identical to its port number, and assigning the same port numbers as Keys to the corresponding Aggregators for each port. A device with this initial configuration will bring up all links as individual, non-aggregated links. The second is achieved by assigning the same administrative and operational Key values to all ports with a common set of capabilities, and also to all Aggregators. A device with this initial configuration will attempt to aggregate together any set of links that have the same Partner System ID and operational Key, and for which both Systems are prepared to allow aggregation.

5.3.6 Link Aggregation Group identification

A Link Aggregation Group consists of either

a) One or more Aggregateable links that terminate in the same pair of Systems and whose ports belong to the same Key Group in each System, or

b) An Individual link.

5.3.6.1 Construction of the Link Aggregation Group Identifier

A unique Link Aggregation Group Identifier (LAG ID) is constructed from the following parameters for each of the communicating Systems:

a) The System Identifier

b) The operational Key assigned to the ports in the LAG

c) The Port Identifier, if the link is identified as an Individual link

The local System's values for these parameters shall be non-zero. In cases where the local System is unable to determine the remote System's values for these parameters by exchange of protocol information, administrative values are used in the construction of the LAG ID. The value of these administrative parameters for the remote System may be configured as zero, provided that the port(s) concerned are also configured to be Individual.

A compound identifier formed from the System Identifiers and Key values alone is sufficient to identify a LAG comprising Aggregateable links. However, such an identifier is not sufficient for a LAG comprising a single Individual link where the Partner System Identifier and operational Key may be zero. Even if these are non-zero there may be multiple Individual Links with the same System Identifier and operational Key combinations, and it becomes necessary to include Port Identifiers to provide unique LAG IDs.

Given that

d) S and T are System Identifiers,

e) K and L are the operational Keys assigned to a LAG by S and T respectively, and

f) P and Q are the Port Identifiers of the ports being attached if the LAG comprises a single Individual Link and zero if the LAG comprises one or more Aggregateable links,

then the general form of the unique LAG ID is [(SKP), (TLQ)].

To simplify comparison of LAG IDs it is conventional to order these such that S is the numerically smaller of S and T.

5.3.6.2 Representation of the Link Aggregation Group Identifier

In order to allow for convenient transcription and interpretation by human network personnel, this standard provides a convention for representing compound LAG IDs. Using this format

a) All fields are written as hexadecimal numbers, two digits per octet, in canonical format.

b) Octets are presented in order, from left to right. Within fields carrying numerical significance (e.g., priority values), the most significant octet is presented first, and the least significant octet last.

c) Within fields that carry MAC addresses, successive octets are separated by dashes (-), in accordance with the hexadecimal representation for MAC addresses defined in IEEE Std 802-1990.

d) Parameters of the LAG ID are separated by commas.

For example, consider the parameters for the two Partners in a Link Aggregation Group shown in Table 5–1.

Table 5–1—Example Partner parameters

	Partner SKP	Partner TLQ
System Parameters (S, T)	System Priority = 0x8000 (see 5.4.2.2) System Identifier = AC-DE-48-03-67-80	System Priority = 0x8000 (see 5.4.2.2) System Identifier = AC-DE-48-03-FF-FF
Key Parameter (K, L)	Key = 0x0001	Key = 0x00AA
Port Parameters (P, Q)	Port Priority = 0x80 (see 5.4.2.2) Port Number = 0x0002	Port Priority = 0x80 (see 5.4.2.2) Port Number = 0x0002

The complete LAG ID derived from this information is represented as follows, for an Individual link:

[(SKP), (TLQ)] = [(8000,AC-DE-48-03-67-80,0001,80,0002), (8000,AC-DE-48-03-FF-FF,00AA,80,0002)]

The corresponding LAG ID for a set of Aggregateable links is represented as follows:

[(SKP), (TLQ)] = [(8000,AC-DE-48-03-67-80,0001,00,0000), (8000,AC-DE-48-03-FF-FF,00AA,00,0000)]

NOTE—The difference between the two representations is that, for an Aggregateable link, the port identifier components are zero.

It is recommended that this format be used whenever displaying LAG ID information for use by network personnel.

5.3.7 Selecting a Link Aggregation Group

Each port is selected for membership in the Link Aggregation Group uniquely identified by the LAG ID (composed of operational information, both derived from local administrative parameters and received through the Link Aggregation Control Protocol). Initial determination of the LAG ID is delayed to allow receipt of such information from a peer Link Aggregation Control entity; in the event such information is not received, locally configured administrative defaults are assumed for the remote port's operational parameters.

Where a particular link is known to be Individual, the complete LAG ID is not required to select the Link Aggregation Group since the link will not be aggregated with any other.

5.3.8 Agreeing on a Link Aggregation Group

Before frames are distributed and collected from a link, both the local Link Aggregation Control entity and its remote peer (if present) need to agree on the Link Aggregation Group. The Link Aggregation Control Protocol allows each of the communicating entities to check their peer's current understanding of the LAG ID, and facilitates rapid exchange of operational parameters while that understanding differs from their own. The protocol entities monitor their operation and, if agreement is not reached (perhaps due to an implementation failure), management is alerted.

The ability of LACP to signal that a particular link is Individual can accelerate the use of the link since, if both Link Aggregation Control entities know that the link is Individual, full agreement on the LAG ID is not necessary.

5.3.9 Attaching a link to an Aggregator

Once a link has selected a Link Aggregation Group, Link Aggregation Control can attach that link to a compatible Aggregator. An Aggregator is compatible if

a) The Aggregator's operational Key matches the port's operational Key, and
b) All other links currently attached to the Aggregator have selected the same Link Aggregation Group.

If several compatible Aggregators exist, Link Aggregation Control may employ a locally determined algorithm, either to ensure deterministic behavior (i.e., independence from the order in which Aggregators become available) or to maximize availability of the aggregation to a MAC Client. If no compatible Aggregator exists, then it is not possible to enable the link until such a time as a compatible Aggregator becomes available.

NOTE—In a properly configured System, there should always be a suitable Aggregator available with the proper Key assigned to serve a newly created Link Aggregation Group, so the unavailability of a compatible Aggregator is normally a temporary state encountered while links are moved between Aggregators. However, given the flexibility of the Key scheme, and given that in some implementations there may not be enough Aggregators to service a given configuration of links, it is possible to create configurations in which there is no Aggregator available to serve a newly identified LAG, in which case the links that are members of that LAG cannot become active until such a time as the configuration is changed to free up an appropriate Aggregator.

Links that are not successful candidates for aggregation (e.g., links that are attached to other devices that cannot perform aggregation or links that have been manually configured to be non-aggregateable) are enabled to operate as individual IEEE 802.3 links. For consistency of modeling, such a link is regarded as being attached to a compatible Aggregator that can only be associated with a single link. That is, from the perspective of Link Aggregation, non-aggregated links are not a special case; they compose an aggregation with a maximum membership of one link.

More than one link can select the same Link Aggregation Group within a short period of time and, as these links detach from their prior Aggregators, additional compatible Aggregators can become available. In order

to avoid such events causing repeated configuration changes, Link Aggregation Control applies hysteresis to the attachment process and allows multiple links to be attached to an Aggregator at the same time.

5.3.10 Signaling readiness to transfer user data

Once a link has been attached to an Aggregator (5.3.9) compatible with the agreed-upon Link Aggregation Group (5.3.8), each Link Aggregation Control entity signals to its peer its readiness to transfer user data to and from the Aggregator's MAC Client. In addition to allowing time for the organization of local Aggregator resources, including the possibility that a compatible Aggregator may not exist, explicit signaling of readiness to transfer user data can be delayed to ensure preservation of frame ordering and prevention of frame duplication. Link Aggregation Control will not signal readiness until it is certain that there are no frames in transit on the link that were transmitted while the link was a member of a previous Link Aggregation Group. This may involve the use of an explicit Marker protocol that ensures that no frames remain to be received at either end of the link before reconfiguration takes place. The operation of the Marker protocol is described in 5.5. The decision as to when, or if, the Marker protocol is used is entirely dependent upon the nature of the distribution algorithm that is employed.

5.3.11 Enabling Collection and Distribution

Every Aggregator can enable or disable Collection and Distribution of frames for each port that is attached to the Aggregator. Initially, both Collection and Distribution are disabled. Once the Link Aggregation Control entity is ready to transfer user data using the link and its peer entity has also signaled readiness, the process of enabling the link can proceed. The Collector is enabled (thus preparing it to receive frames sent over the link by the remote Aggregator's Distributor) and that fact is communicated to the Partner. Once the received information indicates that the remote Aggregator's Collector is enabled, the Distributor is also enabled.

NOTE—This description assumes that the implementation is capable of controlling the state of the transmit and receive functions of the MAC independently. In an implementation where this is not possible, the transmit and receive functions are enabled or disabled together. The manner in which this is achieved is detailed in the description of the Mux machine (see 5.4.15).

If at least one port's Mux in the Link Aggregation Group is Collecting, then the Receive state of the corresponding Aggregator will be Enabled. If at least one port's Mux in the Link Aggregation Group is Distributing, then the Transmit state of the corresponding Aggregator will be Enabled.

5.3.12 Monitoring the membership of a Link Aggregation Group

Each link is monitored in order to confirm that the Link Aggregation Control functions at each end of the link still agree on the configuration information for that link. If the monitoring process detects a change in configuration that materially affects the link's membership in its current LAG, then it may be necessary to remove the link from its current LAG and to move it to a new LAG.

5.3.13 Detaching a link from an Aggregator

A port may be detached from the Aggregator used by its Link Aggregation Group as a result of protocol (e.g., Key) changes, or because of System constraints (e.g., exceeding a maximum allowable number of aggregated links, or device failures) at either end of the link. Both classes of events will cause the LAG ID information for the link to change, and it will be necessary for Link Aggregation Control to detach the link from its current Aggregator and move it to a new LAG (if possible). At the point where the change is detected, the Collecting and Distributing states for the port are set to FALSE. The Frame Distribution function is informed that the link is no longer part of the group, the changed configuration information is communicated to the corresponding Link Aggregation Partner, then the Frame Collection function is informed that the link is no longer part of the group.

Once a link has been removed from its Aggregator, the link can select its new Link Aggregation Group and then attach to a compatible Aggregator, as described in 5.3.7 and 5.3.9.

Any conversation that is reallocated to a different link as a result of detaching a link from an Aggregator shall have its frame ordering preserved. This may involve the use of the Marker protocol to ensure that no frames that form part of that conversation remain to be received at either end of the old link before the conversation can proceed on the new link.

5.3.14 Configuration and administrative control of Link Aggregation

Administrative configuration facilities allow a degree of control to be exerted over the way that links may be aggregated. In particular, administrative configuration allows

a) Key values associated with a port to be identified or modified.
b) Key values associated with an Aggregator to be identified or modified.
c) Links to be identified as being incapable of aggregation.
d) Link Aggregation Control Protocol parameters to be identified or modified.

5.3.15 Link Aggregation Control state information

The Link Aggregation Control function maintains the following information with respect to each link:

a) The identifier of the Link Aggregation Group to which it currently belongs.
b) The identifier of the Aggregator associated with that Link Aggregation Group.
c) The status of interaction between the Frame Collection function of the Aggregator and the link (Collecting TRUE or Collecting FALSE). Collecting TRUE indicates that the receive function of this link is enabled with respect to its participation in an aggregation; i.e., received frames will be passed up to the Aggregator for collection.
d) The status of interaction between the Frame Distribution function of the Aggregator and the link (Distributing TRUE or Distributing FALSE). Distributing TRUE indicates that the transmit function of this link is enabled with respect to its participation in an aggregation; i.e., frames may be passed down from the Aggregator's distribution function for transmission.

This state information is communicated directly between Link Aggregation Control and the Aggregator through shared state variables without the use of a formal service interface.

The Link Aggregation Control function maintains the following information with respect to each Aggregator:

e) The status of the Frame Collection function (Receive Enabled or Receive Disabled).
f) The status of the Frame Distribution function (Transmit Enabled or Transmit Disabled).

These status values are exactly the logical OR of the Collection and Distribution status of the individual links associated with that Aggregator; i.e., if one or more links in the Link Aggregation Group are Collecting, then the Aggregator is Receive Enabled, and if one or more links are Distributing, then the Aggregator is Transmit Enabled.

The Transmit and Receive status of the Aggregator effectively govern the point at which the Aggregator becomes available for use by the MAC Client, or conversely, the point at which it ceases to be available.

5.4 Link Aggregation Control Protocol (LACP)

The Link Aggregation Control Protocol (LACP) provides a standardized means for exchanging information between Partner Systems on a link to allow their Link Aggregation Control instances to reach agreement on

the identity of the Link Aggregation Group to which the link belongs, move the link to that Link Aggregation Group, and enable its transmission and reception functions in an orderly manner.

5.4.1 LACP design elements

The following considerations were taken into account during the development of the protocol described in this subclause:

a) The protocol depends upon the transmission of information and state, rather than the transmission of commands. LACPDUs sent by the first party (the Actor) convey to the second party (the Actor's protocol Partner) what the Actor knows, both about its own state and that of the Partner.

b) The information conveyed in the protocol is sufficient to allow the Partner to determine what action to take next.

c) Active or passive participation in LACP is controlled by LACP_Activity, an administrative control associated with each port, that can take the value Active LACP or Passive LACP. Passive LACP indicates the port's preference for not transmitting LACPDUs unless its Partner's control value is Active LACP (i.e., a preference not to speak unless spoken to). Active LACP indicates the port's preference to participate in the protocol regardless of the Partner's control value (i.e., a preference to speak regardless).

d) Periodic transmission of LACPDUs occurs if the LACP_Activity control of either the Actor or the Partner is Active LACP. These periodic transmissions will occur at either a slow or fast transmission rate depending upon the expressed LACP_Timeout preference (Long Timeout or Short Timeout) of the Partner System.

e) In addition to periodic LACPDU transmissions, the protocol transmits LACPDUs when there is a Need To Transmit (NTT) something to the Partner; i.e., when the Actor's state changes or when it is apparent from the Partner's LACPDUs that the Partner does not know the Actor's current state.

f) The protocol assumes that the rate of LACPDU loss is very low.

There is no explicit frame loss detection/retry mechanism employed by the LACP; however, if information is received from the Partner indicating that it does not have up-to-date information on the Actor's state, or if the next periodic transmission is due, then the Actor will transmit a LACPDU that will correctly update the Partner.

5.4.2 LACPDU structure and encoding

5.4.2.1 Transmission and representation of octets

All LACPDUs comprise an integral number of octets. The bits in each octet are numbered from 0 to 7, where 0 is the low-order bit. When consecutive octets are used to represent a numerical value, the most significant octet is transmitted first, followed by successively less significant octets.

When the encoding of (an element of) a LACPDU is depicted in a diagram

a) Octets are transmitted from top to bottom.

b) Within an octet, bits are shown with bit 0 to the left and bit 7 to the right, and are transmitted from left to right.

c) When consecutive octets are used to represent a binary number, the octet transmitted first has the more significant value.

d) When consecutive octets are used to represent a MAC address, the least significant bit of the first octet is assigned the value of the first bit of the MAC address, the next most significant bit the value of the second bit of the MAC address, and so on through the eighth bit. Similarly, the least significant through most significant bits of the second octet are assigned the value of the ninth through seventeenth bits of the MAC address, and so on for all the octets of the MAC address.

5.4.2.2 LACPDU structure

The LACPDU structure shall be as shown in Figure 5–8 and as further described in the following field definitions:

a) *Destination Address (DA).* The DA in LACPDUs is the Slow_Protocols_Multicast address. Its use and encoding are specified in IEEE Std 802.3 Annex 57A.

b) *Source Address (SA).* The SA in LACPDUs carries the individual MAC address associated with the port through which the LACPDU is transmitted.

c) *Length/Type.* LACPDUs are always Type encoded, and carry the Slow_Protocols_Type field value. The use and encoding of this type is specified in IEEE Std 802.3 Annex 57A.

d) *Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. LACPDUs carry the Subtype value 0x01.

e) *Version Number.* This identifies the LACP version; implementations conformant to this version of the standard carry the value 0x01.

f) *TLV_type = Actor Information.* This field indicates the nature of the information carried in this TLV-tuple. Actor information is identified by the value 0x01.

g) *Actor_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Actor information uses a length value of 20 (0x14).

h) *Actor_System_Priority.* The priority assigned to this System (by management or administration policy), encoded as an unsigned integer.

i) *Actor_System.* The Actor's System ID, encoded as a MAC address.

j) *Actor_Key.* The operational Key value assigned to the port by the Actor, encoded as an unsigned integer.

k) *Actor_Port_Priority.* The priority assigned to this port by the Actor (the System sending the PDU; assigned by management or administration policy), encoded as an unsigned integer.

l) *Actor_Port.* The port number assigned to the port by the Actor (the System sending the PDU), encoded as an unsigned integer.

m) *Actor_State.* The Actor's state variables for the port, encoded as individual bits within a single octet, as follows and as illustrated in Figure 5–9:

   1) *LACP_Activity* is encoded in bit 0. This flag indicates the Activity control value with regard to this link. Active LACP is encoded as a 1; Passive LACP is encoded as a 0.

   2) *LACP_Timeout* is encoded in bit 1. This flag indicates the Timeout control value with regard to this link. Short Timeout is encoded as a 1; Long Timeout is encoded as a 0.

   3) *Aggregation* is encoded in bit 2. If TRUE (encoded as a 1), this flag indicates that the System considers this link to be *Aggregateable*; i.e., a potential candidate for aggregation. If FALSE (encoded as a 0), the link is considered to be *Individual*; i.e., this link can be operated only as an individual link.

   4) *Synchronization* is encoded in bit 3. If TRUE (encoded as a 1), the System considers this link to be *IN_SYNC*; i.e., it has been allocated to the correct Link Aggregation Group, the group has been associated with a compatible Aggregator, and the identity of the Link Aggregation Group is consistent with the System ID and operational Key information transmitted. If FALSE (encoded as a 0), then this link is currently *OUT_OF_SYNC;* i.e., it is not in the right Aggregation.

   5) *Collecting* is encoded in bit 4. TRUE (encoded as a 1) means collection of incoming frames on this link is definitely enabled; i.e., collection is currently enabled and is not expected to be disabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise FALSE (encoded as a 0).

   6) *Distributing* is encoded in bit 5. FALSE (encoded as a 0) means distribution of outgoing frames on this link is definitely disabled; i.e., distribution is currently disabled and is not expected to

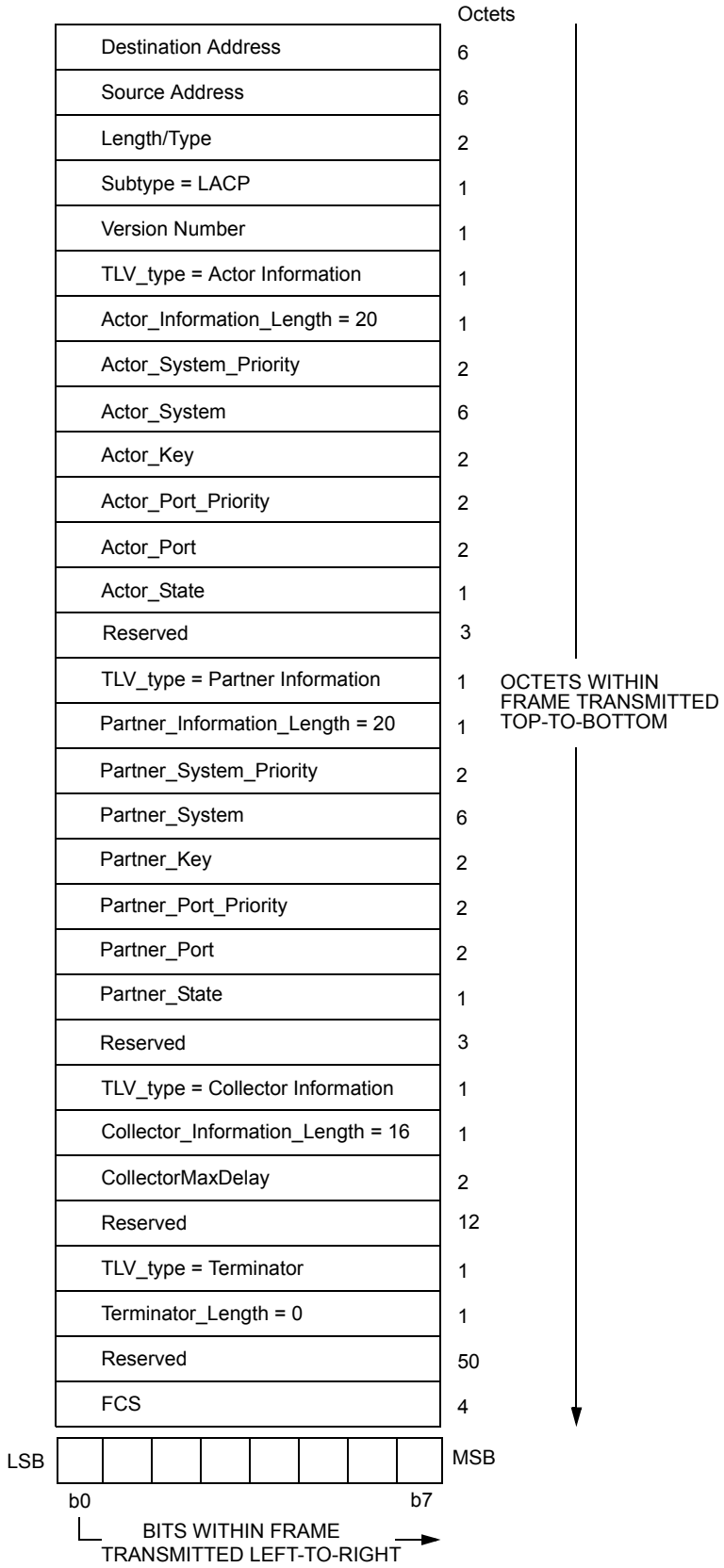


Figure 5–8—LACPDU structure

be enabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise TRUE (encoded as a 1).

7) *Defaulted* is encoded in bit 6. If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is using Defaulted operational Partner information, administratively configured for the Partner. If FALSE (encoded as a 0), the operational Partner information in use has been received in a LACPDU.

8) *Expired* is encoded in bit 7. If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is in the EXPIRED state; if FALSE (encoded as a 0), this flag indicates that the Actor's Receive machine is not in the EXPIRED state.

NOTE—The received values of Defaulted and Expired state are not used by LACP; however, knowing their values can be useful when diagnosing protocol problems.

BIT 0	1	2	3	4	5	6	7
LACP_Activity	LACP_Timeout	Aggregation	Synchronization	Collecting	Distributing	Defaulted	Expired

NOTE—Bit ordering within this field is as specified in 5.4.2.1.

Figure 5–9—Bit encoding of the Actor_State and Partner_State fields

n) *Reserved.* These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeros to claim compliance with Version 1 of this protocol.

o) *TLV_type = Partner Information.* This field indicates the nature of the information carried in this TLV-tuple. Partner information is identified by the integer value 0x02.

p) *Partner_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Partner information uses a length value of 20 (0x14).

q) *Partner_System_Priority.* The priority assigned to the Partner System (by management or administration policy), encoded as an unsigned integer.

r) *Partner_System.* The Partner's System ID, encoded as a MAC address.

s) *Partner_Key.* The operational Key value assigned to the port associated with this link by the Partner, encoded as an unsigned integer.

t) *Partner_Port_Priority.* The priority assigned to this port by the Partner (by management or administration policy), encoded as an unsigned integer.

u) *Partner_Port.* The port number associated with this link assigned to the port by the Partner, encoded as an unsigned integer.

v) *Partner_State.* The Actor's view of the Partner's state variables, depicted in Figure 5–9 and encoded as individual bits within a single octet, as defined for Actor_State.

w) *Reserved.* These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeros to claim compliance with Version 1 of this protocol.

x) *TLV_type = Collector Information.* This field indicates the nature of the information carried in this TLV-tuple. Collector information is identified by the integer value 0x03.

y) *Collector_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Collector information uses a length value of 16 (0x10).

z) *CollectorMaxDelay.* This field contains the value of CollectorMaxDelay (5.2.3.1.1) of the station transmitting the LACPDU, encoded as an unsigned integer number of tens of microseconds. The range of values for this parameter is 0 to 65 535 tens of microseconds (0.65535 s).

aa) *Reserved.* These 12 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeros to claim compliance with Version 1 of this protocol.

ab) *TLV_type = Terminator.* This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information is identified by the integer value 0x00.

ac) *Terminator_Length.* This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

NOTE—The use of a Terminator_Length of 0 is intentional. In TLV encoding schemes it is common practice for the terminator encoding to be 0 both for the type and the length.

ad) *Reserved*. These 50 octets are reserved for use in future extensions to the protocol. They are ignored on receipt and are transmitted as zeros to claim compliance with Version 1 of this protocol.

NOTE—The Reserved octets are included in all valid LACPDUs in order to force the TLV lengths to multiples of 4 octets, and to force a fixed PDU size of 128 octets, regardless of the version of the protocol. Hence, a Version 1 implementation is guaranteed to be able to receive version N PDUs successfully, although version N PDUs may contain additional information that cannot be interpreted (and will be ignored) by the Version 1 implementation. A crucial factor in ensuring backwards compatibility is that any future version of the protocol is required not to redefine the structure or semantics of information defined for the previous version; it may only add new information elements to the previous set. Hence, in a version N PDU, a Version 1 implementation can expect to find the Version 1 information in exactly the same places as in a Version 1 PDU, and can expect to interpret that information as defined for Version 1. Future versions of this protocol expect to have the Reserved octets available for their use.

ae) *FCS.* This field is the Frame Check Sequence, typically generated by the underlying MAC.

5.4.3 LACP state machine overview

The operation of the protocol is controlled by a number of state machines, each of which performs a distinct function. These state machines are for the most part described on a per-port basis; any deviations from per-port description are highlighted in the text. Events (such as expiration of a timer or received LACPDUs) may cause state transitions and also cause actions to be taken; those actions may include the need for transmission of a LACPDU containing repeated or new information. Periodic and event-driven transmissions are controlled by the state of a Need-To-Transmit (NTT) variable (see 5.4.7), generated by the state machines as necessary.

The state machines are as follows:

a) *Receive machine (RX—5.4.12).* This state machine receives LACPDUs from the Partner, records the information contained, and times it out using either Short Timeouts or Long Timeouts, according to the setting of LACP_Timeout. It evaluates the incoming information from the Partner to determine whether the Actor and Partner have both agreed upon the protocol information exchanged to the extent that the port can now be safely used, either in an aggregation with other ports or as an individual port; if not, it asserts NTT in order to transmit fresh protocol information to the Partner. If the protocol information from the Partner times out, the Receive machine installs default parameter values for use by the other state machines.

b) *Periodic Transmission machine (5.4.13)*. This state machine determines whether the Actor and its Partner will exchange LACPDUs periodically in order to maintain an aggregation (periodic LACPDU exchanges occur if either or both are configured for Active LACP).

c) *Selection Logic (5.4.14)*. The Selection Logic is responsible for selecting the Aggregator to be associated with this port.

d) *Mux machine (MUX—5.4.15).* This state machine is responsible for attaching the port to a selected Aggregator, detaching the port from a de-selected Aggregator, and for turning collecting and distributing at the port on or off as required by the current protocol information.

e) *Transmit machine (TX—5.4.16)*. This state machine handles the transmission of LACPDUs, both on demand from the other state machines, and on a periodic basis.

Figure 5–10 illustrates the relationships among these state machines and the flow of information between them. The set of arrows labeled Partner State Information represents new Partner information, contained in an incoming LACPDU or supplied by administrative default values, being fed to each state machine by the Receive machine. The set of arrows labeled Actor State Information represents the flow of updated Actor state information between the state machines. Transmission of LACPDUs occurs either as a result of the Periodic machine determining the need to transmit a periodic LACPDU, or as a result of changes to the Actor's state information that need to be communicated to the Partner. The need to transmit a LACPDU is signaled to the Transmit machine by asserting NTT. The remaining arrows represent shared variables in the state machine description that allow a state machine to cause events to occur in another state machine.

NOTE—The arrows marked Ready_N show that information derived from the operation of another port or ports can affect the operation of a port's state machine. See the definition of the Ready and Ready_N variables in 5.4.8.

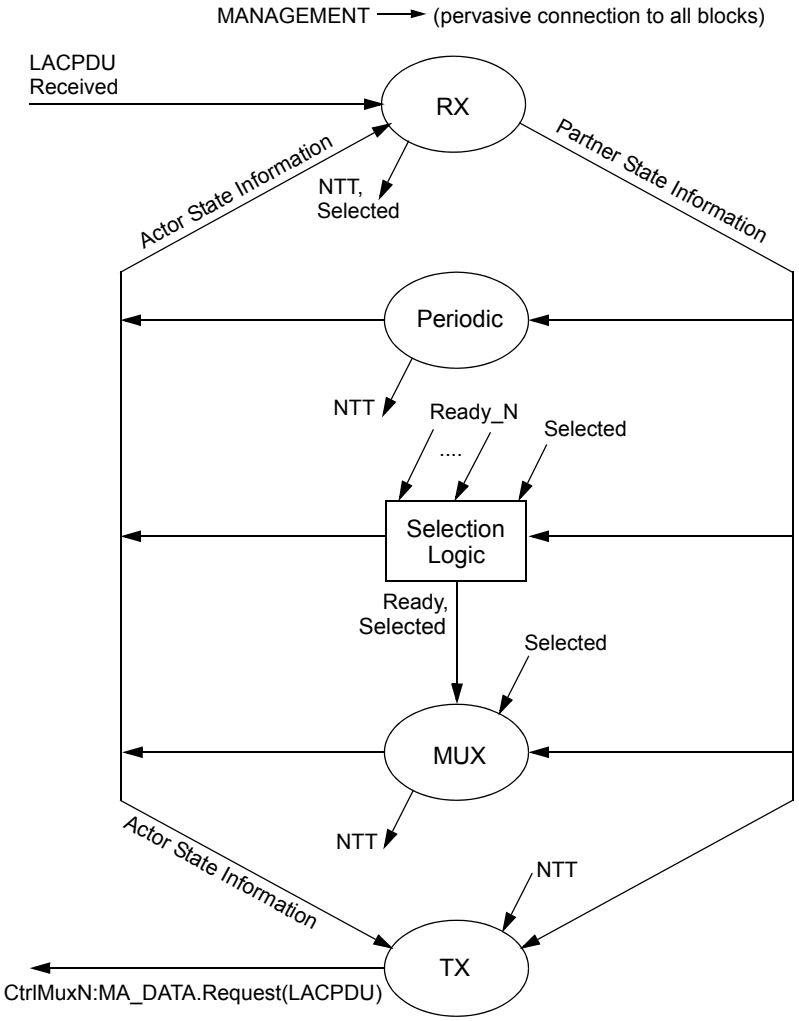


Figure 5–10—Interrelationships among state machines

Two further state machines are defined for diagnostic purposes. They are as follows:

f) *Actor and Partner Churn Detection machines (5.4.17).* These state machines make use of the IN_SYNC and OUT_OF_SYNC states generated by the Actor and Partner Mux machines in order to detect the situation where the state machines are unable to resolve the state of a given link; e.g., because the Partner System repeatedly sends conflicting information in its LACPDUs. As this

situation can occur in a normally functioning link, particularly where either or both participating Systems have constrained aggregation capability (see 5.6), these state machines simply detect the presence of such a condition and signal its existence to management.

5.4.4 Constants

All timers specified in this subclause have an implementation tolerance of ± 250 ms.

Fast_Periodic_Time

The number of seconds between periodic transmissions using Short Timeouts.

Value: Integer

1

Slow_Periodic_Time

The number of seconds between periodic transmissions using Long Timeouts.

Value: Integer

30

Short_Timeout_Time

The number of seconds before invalidating received LACPDU information when using Short Timeouts (3 × Fast_Periodic_Time).

Value: Integer

3

Long_Timeout_Time

The number of seconds before invalidating received LACPDU information when using Long Timeouts (3 × Slow_Periodic_Time).

Value: Integer

90

Churn_Detection_Time

The number of seconds that the Actor and Partner Churn state machines wait for the Actor or Partner Sync state to stabilize.

Value: Integer

60

Aggregate_Wait_Time

The number of seconds to delay aggregation, to allow multiple links to aggregate simultaneously.

Value: Integer

2

5.4.5 Variables associated with the System

Actor_System

The MAC address component of the System Identifier of the System.

Value: 48 bits

Assigned by administrator or System policy.

Actor_System_Priority

The System Priority of the System.

Value: Integer

Assigned by administrator or System policy.

5.4.6 Variables associated with each Aggregator

Aggregator_MAC_address

The MAC address assigned to the Aggregator.

Value: 48 bits

Assigned by administrator or System policy.

Aggregator_Identifier

Used to uniquely identify an Aggregator within a System.

Value: Integer

Assigned by administrator or System policy.

Individual_Aggregator

The aggregation capability of the Aggregator.

Value: Boolean

TRUE if the port attached to this Aggregator is not capable of aggregation with any other port.

FALSE if the port(s) attached to this Aggregator are capable of aggregation with other ports.

Actor_Admin_Aggregator_Key

The administrative Key value associated with the Aggregator.

Value: Integer

Assigned by administrator or System policy.

Actor_Oper_Aggregator_Key

The operational Key value associated with the Aggregator.

Value: Integer

Assigned by the Actor.

Partner_System

The MAC address component of the System Identifier of the remote System to which the Aggregator is connected. If the Aggregator has no attached ports, this variable is set to 0x00-00-00-00-00-00.

Value: 48 bits

Partner_System_Priority

The System Priority of the remote System to which the Aggregator is connected. If the Aggregator has no attached ports, this variable is set to zero.

Value: Integer

Partner_Oper_Aggregator_Key

The operational Key assigned to an aggregation by the remote System to which this Aggregator is connected. If the Aggregator has no attached ports, this variable is set to zero.

Value: Integer

Receive_State

The Receive_State of the Aggregator will be Enabled if one or more ports attached to the Aggregator are Collecting (i.e., Actor_Oper_Port_State.Collecting is TRUE for any port). Otherwise, Receive_State is Disabled.

Values: Enabled or Disabled

Transmit_State

The Transmit_State of the Aggregator will be Enabled if one or more ports attached to the Aggregator are Distributing (i.e., Actor_Oper_Port_State.Distributing is TRUE for any port). Otherwise, Transmit_State is Disabled.

Values: Enabled or Disabled

LAG_Ports

The set of ports that belong to the Link Aggregation Group.

Value: Integer Array

5.4.7 Variables associated with each port

Actor_Port_Number

The port number assigned to the port.

Value: Integer

Assigned by administrator or System policy.

Actor_Port_Priority

The priority value assigned to the port, used to converge dynamic Key changes.

Value: Integer

Assigned by administrator or System policy.

Actor_Port_Aggregator_Identifier

The identifier of the Aggregator to which this port is attached.

Value: Integer

NTT

Need To Transmit flag.

Value: Boolean

TRUE indicates that there is new protocol information that should be transmitted on the link, or that the Partner needs to be reminded of the old information.

FALSE otherwise.

Actor_Admin_Port_Key

The administrative value of Key assigned to this port by administrator or System policy.

Value: Integer

Actor_Oper_Port_Key

The operational value of Key assigned to this port by the Actor.

Value: Integer

Actor_Admin_Port_State

The administrative values of the Actor's state parameters. This consists of the following set of variables, as described in 5.4.2.2:

LACP_Activity
LACP_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired

Value: 8 bits

Actor_Oper_Port_State

The operational values of the Actor's state parameters. This consists of the following set of variables, as described in 5.4.2.2:

LACP_Activity
LACP_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired

Value: 8 bits

Partner_Admin_System

Default value for the MAC address component of the System Identifier of the Partner, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: 48 bits

Partner_Oper_System

The operational value of the MAC address component of the System Identifier of the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner_Admin_System.

Value: 48 bits

Partner_Admin_System_Priority

Default value for the System Priority component of the System Identifier of the Partner, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

Partner_Oper_System_Priority

The operational value of the System Priority of the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner_Admin_System_Priority.

Value: Integer

Partner_Admin_Key

Default value for the Partner's Key, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

Partner_Oper_Key

The operational value of the Key value assigned to this link by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner_Admin_Key.

Value: Integer

Partner_Admin_Port_Number

Default value for the Port Number component of the Partner's Port Identifier, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

Partner_Oper_Port_Number

The operational value of the port number assigned to this link by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner_Admin_Port_Number.

Value: Integer

Partner_Admin_Port_Priority

Default value for the Port Priority component of the Partner's Port Identifier, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

Partner_Oper_Port_Priority

The operational value of the priority value assigned to this link by the Partner, used to converge dynamic Key changes. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner_Admin_Port_Priority.

Value: Integer

Partner_Admin_Port_State

Default value for the Partner's state parameters, assigned by administrator or System policy for use when the Partner's information is unknown or expired. The value consists of the following set of variables, as described in 5.4.2.2:

LACP_Activity

LACP_Timeout

Aggregation

Synchronization

Collecting

Distributing
Defaulted
Expired

The value of Collecting shall be set the same as the value of Synchronization.

Value: 8 bits

Partner_Oper_Port_State

The operational value of the Actor's view of the current values of the Partner's state parameters. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner_Admin_Port_State. The value consists of the following set of variables, as described in 5.4.2.2:

LACP_Activity
LACP_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired

Value: 8 bits

port_enabled

A variable indicating that the physical layer has indicated that the link has been established and the port is operable.

Value: Boolean

TRUE if the physical layer has indicated that the port is operable.

FALSE otherwise.

NOTE—The means by which the value of the port_enabled variable is generated by the underlying MAC and/or PHY is implementation-dependent.

5.4.8 Variables used for managing the operation of the state machines

BEGIN

This variable indicates the initialization (or reinitialization) of the LACP protocol entity. It is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.

Value: Boolean

LACP_Enabled

This variable indicates that the port is operating the LACP. If the port is operating in half duplex, the value of LACP_Enabled shall be FALSE. Otherwise, the value of LACP_Enabled shall be TRUE.

Value: Boolean

actor_churn

This variable indicates that the Actor Churn Detection machine has detected that a local port configuration has failed to converge within a specified time, and that management intervention is required.

Value: Boolean

partner_churn

This variable indicates that the Partner Churn Detection machine has detected that a remote port configuration has failed to converge within a specified time, and that management intervention is required.

Value: Boolean

Ready_N

The port asserts Ready_N TRUE to indicate to the Selection Logic that the wait_while_timer has expired and it is waiting (i.e., the port is in the WAITING state) to attach to an Aggregator. Otherwise, its value is FALSE. There is one Ready_N value for each port.

Value: Boolean

Ready

The Selection Logic asserts Ready TRUE when the values of Ready_N for all ports that are waiting to attach to a given Aggregator are TRUE. If any of the values of Ready_N for the ports that are waiting to attach to that Aggregator are FALSE, or if there are no ports waiting to attach to that Aggregator, then the value of Ready is FALSE.

Value: Boolean

Selected

A value of SELECTED indicates that the Selection Logic has selected an appropriate Aggregator. A value of UNSELECTED indicates that no Aggregator is currently selected. A value of STANDBY indicates that although the Selection Logic has selected an appropriate Aggregator, aggregation restrictions currently prevent the port from being enabled as part of the aggregation, and so the port is being held in a standby condition. This variable can only be set to SELECTED or STANDBY by the operation of the port's Selection Logic. It can be set to UNSELECTED by the operation of the port's Receive machine, or by the operation of the Selection Logic associated with another port.

NOTE—Setting Selected UNSELECTED in the Selection Logic associated with another port occurs if the Selection Logic determines that the other port has a stronger claim to attach to this port's current Aggregator.

Value: SELECTED, UNSELECTED, or STANDBY

port_moved

This variable is set to TRUE if the Receive machine for a port is in the PORT_DISABLED state, and the combination of Partner_Oper_System and Partner_Oper_Port_Number in use by that port has been received in an incoming LACPDU on a different port. This variable is set to FALSE once the INITIALIZE state of the Receive machine has set the Partner information for the port to administrative default values.

Value: Boolean

5.4.9 Functions

recordPDU

This function records the parameter values for the Actor carried in a received LACPDU (Actor_Port, Actor_Port_Priority, Actor_System, Actor_System_Priority, Actor_Key, and Actor_State variables) as the current Partner operational parameter values (Partner_Oper_Port_Number, Partner_Oper_Port_Priority, Partner_Oper_System,

Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State variables with the exception of Synchronization) and sets Actor_Oper_Port_State.Defaulted to FALSE.

This function also updates the value of the Partner_Oper_Port_State.Synchronization using the parameter values carried in received LACPDUs. Parameter values for the Partner carried in the received PDU (Partner_Port, Partner_Port_Priority, Partner_System, Partner_System_Priority, Partner_Key, and Partner_State.Aggregation) are compared to the corresponding operational parameter values for the Actor (Actor_Port_Number, Actor_Port_Priority, Actor_System, Actor_System_Priority, Actor_Oper_Port_Key, and Actor_Oper_Port_State.Aggregation). Partner_Oper_Port_State.Synchronization is set to TRUE if all of these parameters match, Actor_State.Synchronization in the received PDU is set to TRUE, and LACP will actively maintain the link in the aggregation.

Partner_Oper_Port_State.Synchronization is also set to TRUE if the value of Actor_State.Aggregation in the received PDU is set to FALSE (i.e., indicates an Individual link), Actor_State.Synchronization in the received PDU is set to TRUE, and LACP will actively maintain the link.

Otherwise, Partner_Oper_Port_State.Synchronization is set to FALSE.

LACP is considered to be actively maintaining the link if either the PDU's Actor_State.LACP_Activity variable is TRUE or both the Actor's Actor_Oper_Port_State.LACP_Activity and the PDU's Partner_State.LACP_Activity variables are TRUE.

recordDefault

This function records the default parameter values for the Partner carried in the Partner Admin parameters (Partner_Admin_Port_Number, Partner_Admin_Port_Priority, Partner_Admin_System, Partner_Admin_System_Priority, Partner_Admin_Key, and Partner_Admin_Port_State) as the current Partner operational parameter values (Partner_Oper_Port_Number, Partner_Oper_Port_Priority, Partner_Oper_System, Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State) and sets Actor_Oper_Port_State.Defaulted to TRUE.

update_Selected

This function updates the value of the Selected variable, using parameter values from a newly received LACPDU. The parameter values for the Actor carried in the received PDU (Actor_Port, Actor_Port_Priority, Actor_System, Actor_System_Priority, Actor_Key, and Actor_State.Aggregation) are compared with the corresponding operational parameter values for the port's Partner (Partner_Oper_Port_Number, Partner_Oper_Port_Priority, Partner_Oper_System, Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State.Aggregation). If one or more of the comparisons show that the value(s) received in the PDU differ from the current operational values, then Selected is set to UNSELECTED. Otherwise, Selected remains unchanged.

update_Default_Selected

This function updates the value of the Selected variable, using the Partner administrative parameter values. The administrative values (Partner_Admin_Port_Number, Partner_Admin_Port_Priority, Partner_Admin_System, Partner_Admin_System_Priority, Partner_Admin_Key, and Partner_Admin_Port_State.Aggregation) are compared with the corresponding operational parameter values for the Partner (Partner_Oper_Port_Number,

Partner_Oper_Port_Priority, Partner_Oper_System, Partner_Oper_System_Priority, Partner_Oper_Key, and Partner_Oper_Port_State.Aggregation). If one or more of the comparisons shows that the administrative value(s) differ from the current operational values, then Selected is set to UNSELECTED. Otherwise, Selected remains unchanged.

update_NTT

This function updates the value of the NTT variable, using parameter values from a newly received LACPDU. The parameter values for the Partner carried in the received PDU (Partner_Port, Partner_Port_Priority, Partner_System, Partner_System_Priority, Partner_Key, Partner_State.LACP_Activity, Partner_State.LACP_Timeout, Partner_State.Synchronization, and Partner_State.Aggregation) are compared with the corresponding operational parameter values for the Actor (Actor_Port_Number, Actor_Port_Priority, Actor_System, Actor_System_Priority, Actor_Oper_Port_Key, Actor_Oper_Port_State.LACP_Activity, Actor_Oper_Port_State.LACP_Timeout, Actor_Oper_Port_State.Synchronization, and Actor_Oper_Port_State.Aggregation). If one or more of the comparisons show that the value(s) received in the PDU differ from the current operational values, then NTT is set to TRUE. Otherwise, NTT remains unchanged.

Attach_Mux_To_Aggregator

This function causes the port's Control Parser/Multiplexer to be attached to the Aggregator Parser/Multiplexer of the selected Aggregator, in preparation for collecting and distributing frames.

Detach_Mux_From_Aggregator

This function causes the port's Control Parser/Multiplexer to be detached from the Aggregator Parser/Multiplexer of the Aggregator to which the port is currently attached.

Enable_Collecting

This function causes the Aggregator Parser of the Aggregator to which the port is attached to start collecting frames from the port.

Disable_Collecting

This function causes the Aggregator Parser of the Aggregator to which the port is attached to stop collecting frames from the port.

Enable_Distributing

This function causes the Aggregator Multiplexer of the Aggregator to which the port is attached to start distributing frames to the port.

Disable_Distributing

This function causes the Aggregator Multiplexer of the Aggregator to which the port is attached to stop distributing frames to the port.

Enable_Collecting_Distributing

This function causes the Aggregator Parser of the Aggregator to which the port is attached to start collecting frames from the port, and the Aggregator Multiplexer to start distributing frames to the port.

Disable_Collecting_Distributing

This function causes the Aggregator Parser of the Aggregator to which the port is attached to stop collecting frames from the port, and the Aggregator Multiplexer to stop distributing frames to the port.

5.4.10 Timers

current_while_timer

This timer is used to detect whether received protocol information has expired. If Actor_Oper_State.LACP_Timeout is set to Short Timeout, the timer is started with the value Short_Timeout_Time. Otherwise, it is started with the value Long_Timeout_Time (see 5.4.4).

actor_churn_timer

This timer is used to detect Actor churn states. It is started using the value Churn_Detection_Time (see 5.4.4).

periodic_timer (time_value)

This timer is used to generate periodic transmissions. It is started using the value Slow_Periodic_Time or Fast_Periodic_Time (see 5.4.4), as specified in the Periodic Transmission state machine.

partner_churn_timer

This timer is used to detect Partner churn states. It is started using the value Churn_Detection_Time (see 5.4.4).

wait_while_timer

This timer provides hysteresis before performing an aggregation change, to allow all links that will join this Aggregation to do so. It is started using the value Aggregate_Wait_Time (see 5.4.4).

5.4.11 Messages

CtrlMuxN:MA_DATA.indication(LACPDU)

This message is generated by the Control Parser as a result of the reception of a LACPDU, formatted as defined in 5.4.2.

5.4.12 Receive machine

The Receive machine shall implement the function specified in Figure 5–11 with its associated parameters (5.4.4 through 5.4.11).

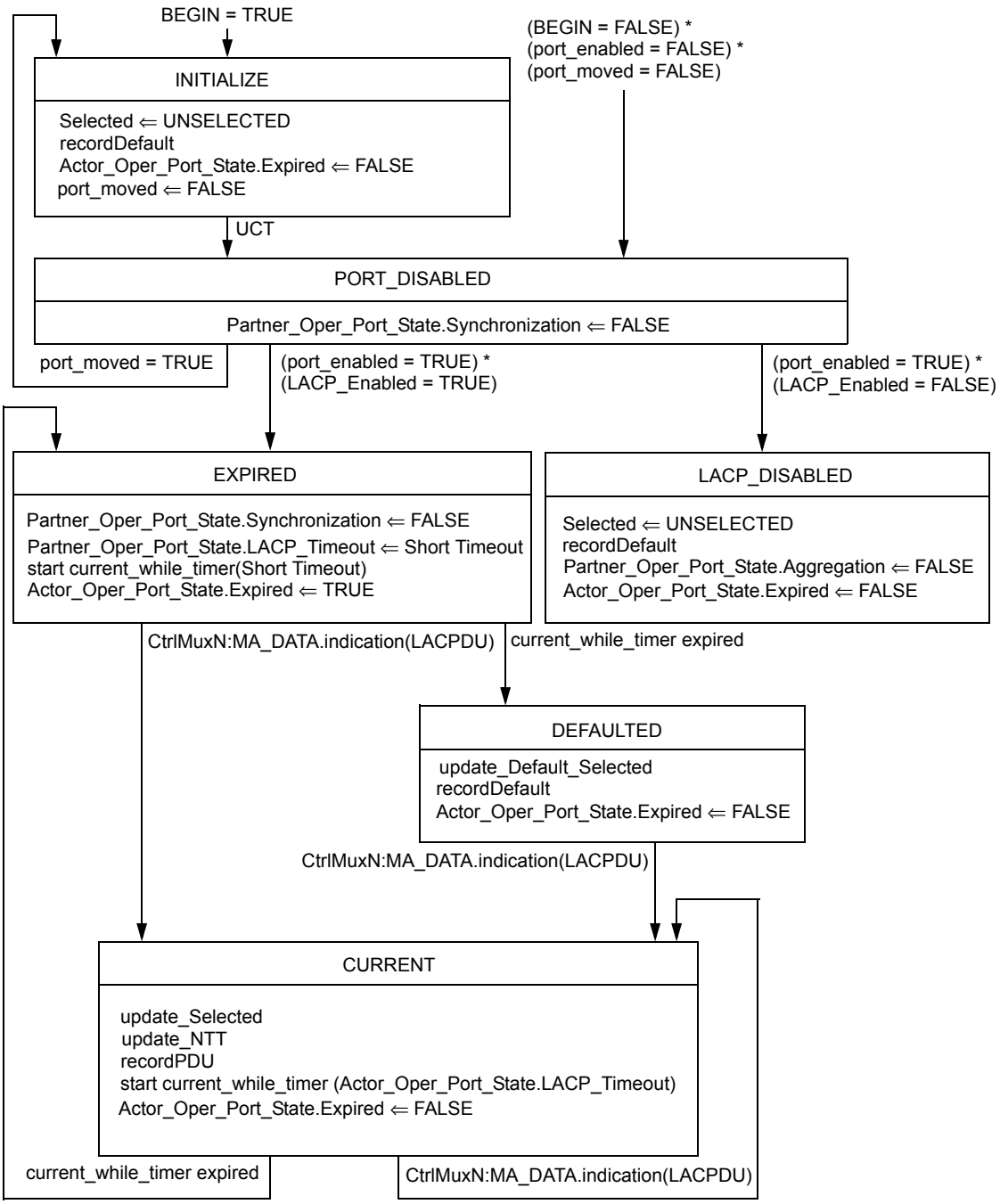


Figure 5–11—Receive machine state diagram

On receipt of a LACPDU, the state machine enters the CURRENT state. The update_Selected function sets the Selected variable UNSELECTED if the Link Aggregation Group identified by the combination of the protocol Partner's own information and the Actor's own information has changed. The Selected variable is used by the Mux machine (5.4.15).

NOTE 1—The Receive machine may set the Selected variable to UNSELECTED; however, setting this variable to SELECTED or STANDBY is the responsibility of the Selection Logic.

The update_NTT function is used to determine whether further protocol transmissions are required; NTT is set to TRUE if the Partner's view of the Actor's operational parameters is not up to date. The recordPDU function records the information contained in the LACPDU in the Partner operational variables, and the current_while timer is started. The value used to start the timer is either Short_Timeout_Time or Long_Timeout_Time, depending upon the Actor's operational value of LACP_Timeout.

In the process of executing the recordPDU function, a Receive machine compliant to this standard shall not validate the Version Number, TLV_type, or Reserved fields in received LACPDUs. The same actions are taken regardless of the values received in these fields. A Receive machine may validate the Actor_Information_Length, Partner_Information_Length, Collector_Information_Length, or Terminator_Length fields. These behaviors, together with the constraint on future protocol enhancements, are discussed in 5.4.2.2.

NOTE 2—The rules expressed above allow Version 1 devices to be compatible with future revisions of the protocol.

If no LACPDU is received before the current_while timer expires, the state machine transits to the EXPIRED state. The Partner_Oper_Port_State.Synchronization variable is set to FALSE, the current operational value of the Partner's LACP_Timeout variable is set to Short Timeout, and the current_while timer is started with a value of Short_Timeout_Time. This is a transient state; the LACP_Timeout settings allow the Actor to transmit LACPDUs rapidly in an attempt to re-establish communication with the Partner.

If no LACPDU is received before the current_while timer expires again, the state machine transits to the DEFAULTED state. The recordDefault function overwrites the current operational parameters for the Partner with administratively configured values. This allows configuration of aggregations and individual links when no protocol Partner is present, while still permitting an active Partner to override default settings. The update_Default_Selected function sets the Selected variable UNSELECTED if the Link Aggregation Group has changed. Since all operational parameters are now set to locally administered values, there can be no disagreement as to the Link Aggregation Group, so the Partner_Oper_Port_State.Synchronization variable is set to TRUE.

If the port becomes inoperable and a BEGIN event has not occurred, the state machine enters the PORT_DISABLED state. Partner_Oper_Port_State.Synchronization is set to FALSE. This state allows the current Selection state to remain undisturbed, so that, in the event that the port is still connected to the same Partner and Partner port when it becomes operable again, there will be no disturbance caused to higher layers by unneccessary re-configuration. If the same Actor System ID and Port are seen in a LACPDU received on a different Port (port_moved is set to TRUE), this indicates that the physical connectivity has changed, and causes the state machine to enter the INITIALIZE state. This state is also entered if a BEGIN event occurs.

The INITIALIZE state causes the administrative values of the Partner parameters to be used as the current operational values, and sets Selected to UNSELECTED. These actions force the Mux machine to detach the port from its current Aggregator. The variable port_moved is set to FALSE; if the entry to INITIALIZE occurred as a result of port_moved being set to TRUE, then the state machine will immediately transition back to the PORT_DISABLED state.

If the port is operating in half duplex, the operation of LACP is disabled on the port (LACP_Enabled is FALSE) and the LACP_DISABLED state is entered. This state is entered following a BEGIN or Port

Enabled event. This state is similar to the DEFAULTED state, except that the port is forced to operate as an Individual port, as the value of Partner_Oper_Port_State.Aggregation is forced to Individual. Exit from this state occurs on a BEGIN or Port Disabled event.

5.4.13 Periodic Transmission machine

The Periodic Transmission machine shall implement the function specified in Figure 5–12 with its associated parameters (5.4.4 through 5.4.11).

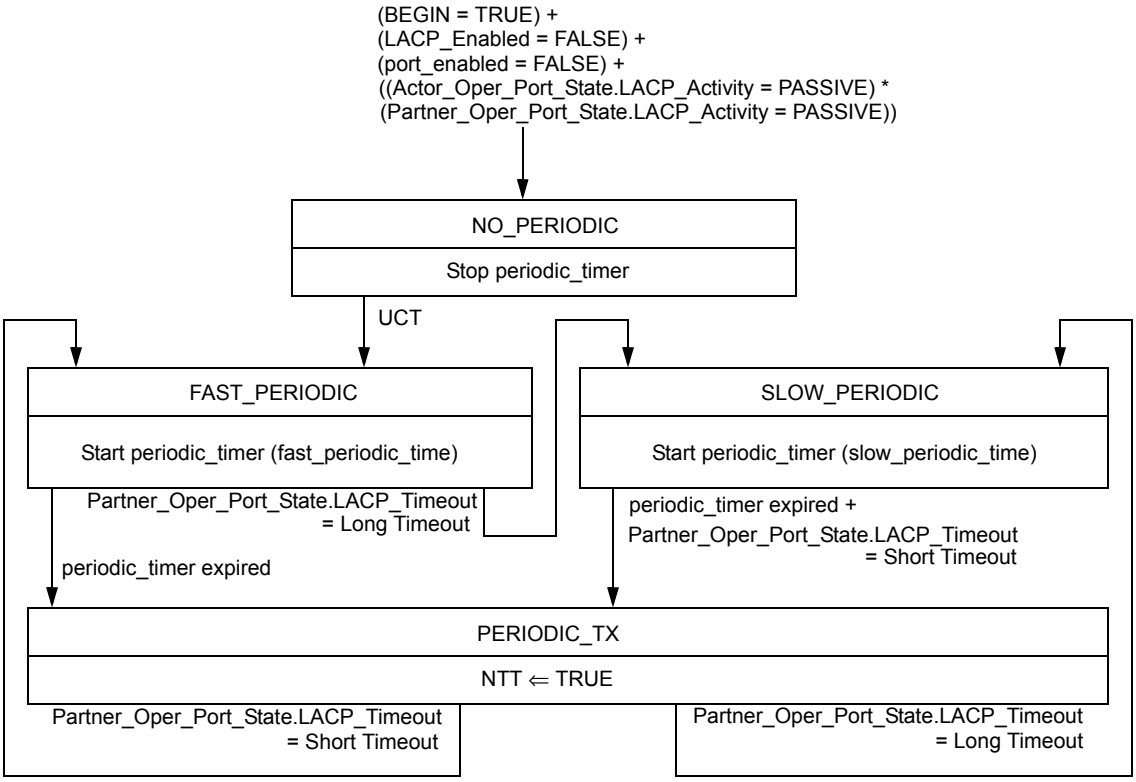


Figure 5–12—Periodic Transmission machine state diagram

The Periodic Transmission machine establishes the desire of the Actor and Partner to exchange periodic LACPDUs on the link in order to maintain an aggregation, and establishes how often those periodic transmissions should occur. Periodic transmissions will take place if either participant so wishes. Transmissions occur at a rate determined by the Partner; this rate is linked to the speed at which the Partner will time out received information.

The state machine has four states. They are as follows:

a) *NO_PERIODIC.* While in this state, periodic transmissions are disabled.

b) *FAST_PERIODIC.* While in this state, periodic transmissions are enabled at a fast transmission rate.

c) *SLOW_PERIODIC.* While in this state, periodic transmissions are enabled at a slow transmission rate.

d) *PERIODIC_TX.* This is a transitory state entered on periodic_timer expiry, that asserts NTT and then exits to FAST_PERIODIC or SLOW_PERIODIC depending upon the Partner's LACP_Timeout setting.

The values of Partner_Oper_Port_State.LACP_Activity and Actor_Oper_Port_State.LACP_Activity determine whether periodic transmissions take place. If either or both parameters are set to Active LACP,

then periodic transmissions occur; if both are set to Passive LACP, then periodic transmissions do not occur. Similarly, if either of the LACP_Enabled or port_enabled variables is set to FALSE, indicating that LACP has been disabled on the port or that the port is non-operational, then no periodic transmissions take place.

If periodic transmissions are enabled, the rate at which they take place is determined by the value of the Partner_Oper_Port_State.LACP_Timeout variable. If this variable is set to Short Timeout, then the value fast_periodic_time is used to determine the time interval between periodic transmissions. Otherwise, slow_periodic_time is used to determine the time interval.

5.4.14 Selection Logic

The Selection Logic selects a compatible Aggregator for a port, using the port's LAG ID. The Selection Logic may determine that the link should be operated as a standby link if there are constraints on the simultaneous attachment of ports that have selected the same Aggregator.

NOTE 1—There will never be more than one Aggregator with the same LAG ID, but there may be none. Normally, the latter will be a temporary state, caused by the fact that it takes a finite time for ports to be moved to the correct Aggregators during reconfiguration.

The Mux machine controls the process of attaching the port to a selected Aggregator, after first detaching the port from any prior Aggregator if the port's LAG ID has changed.

NOTE 2—A port is always detached from its prior Aggregator when the LAG ID changes, even if the same Aggregator is selected later; to do otherwise would be to risk misdelivery of frames. Selection of a new Aggregator cannot take place until the port is detached from any prior Aggregator; other Aggregators may become free while the port is detaching, and other ports may attach to some of the available Aggregators during this time interval.

The operation of the Selection Logic is separated into the following two subclauses:

a) The requirements for the correct operation of the Selection Logic are defined in 5.4.14.1.
b) The recommended default operation of the Selection Logic is described in 5.4.14.2.

This separation reflects the fact that a wide choice of selection rules is possible within the proper operation of the protocol. An implementation that claims conformance to this standard may support selection rules other than the recommended default; however, any such rules shall meet the requirements stated in 5.4.14.1.

5.4.14.1 Selection Logic—Requirements

Aggregation is represented by a port selecting an appropriate Aggregator, and then attaching to that Aggregator. The following are required for correct operation of the selection and attachment logic:

a) The implementation shall support at least one Aggregator per System.
b) Each port shall be assigned an operational Key (5.3.5). Ports that can aggregate together are assigned the same operational Key as the other ports with which they can aggregate; ports that cannot aggregate with any other port are allocated unique operational Keys.
c) Each Aggregator shall be assigned an operational Key.
d) Each Aggregator shall be assigned an identifier that distinguishes it among the set of Aggregators in the System.
e) A port shall only select an Aggregator that has the same operational Key assignment as its own operational Key.
f) Subject to the exception stated in item g), ports that are members of the same Link Aggregation Group (i.e., two or more ports that have the same Actor System ID, Actor Key, Partner System ID, and Partner Key, and that are not required to be Individual) shall select the same Aggregator.
g) Any pair of ports that are members of the same Link Aggregation Group, but are connected together by the same link, shall not select the same Aggregator (i.e., if a loopback condition exists between two ports, they shall not be aggregated together. For both ports, the Actor System ID is the same as

the Partner System ID; also, for port A, the Partner's port identifier is port B, and for port B, the Partner's port identifier is port A).

NOTE—This exception condition prevents the formation of an aggregated link, comprising two ends of the same physical link aggregated together, in which all frames transmitted through an Aggregator are immediately received through the same Aggregator. However, it permits the aggregation of multiple links that are in loopback; for example, if port A is looped back to port C and port B is looped back to port D, then it is permissible for A and B (or A and D) to aggregate together, and for C and D (or B and C) to aggregate together.

h) Any port that is required to be Individual (i.e., the operational state for the Actor or the Partner indicates that the port is Individual) shall not select the same Aggregator as any other port.

i) Any port that is Aggregateable shall not select an Aggregator to which an Individual port is already attached.

j) If the above conditions result in a given port being unable to select an Aggregator, then that port shall not be attached to any Aggregator.

k) If there are further constraints on the attachment of ports that have selected an Aggregator, those ports may be selected as standby in accordance with the rules specified in 5.6.1. Selection or deselection of that Aggregator can cause the Selection Logic to re-evaluate the ports to be selected as standby.

l) The Selection Logic operates upon the operational information recorded by the Receive state machine, along with knowledge of the Actor's own operational configuration and state. The Selection Logic uses the LAG ID for the port, determined from these operational parameters, to locate the correct Aggregator to which to attach the port.

m) The Selection Logic is invoked whenever a port is not attached to and has not selected an Aggregator, and executes continuously until it has determined the correct Aggregator for the port.

NOTE—The Selection Logic may take a significant time to complete its determination of the correct Aggregator, as a suitable Aggregator may not be immediately available, due to configuration restrictions or the time taken to re-allocate ports to other Aggregators.

n) Once the correct Aggregator has been determined, the variable Selected shall be set to SELECTED or to STANDBY (5.4.8, 5.6.1).

NOTE—If Selected is SELECTED, the Mux machine will start the process of attaching the port to the selected Aggregator. If Selected is STANDBY, the Mux machine holds the port in the WAITING state, ready to be attached to its Aggregator once its Selected state changes to SELECTED.

o) The Selection Logic is responsible for computing the value of the Ready variable from the values of the Ready_N variable(s) associated with the set of ports that are waiting to attach to the same Aggregator (see 5.4.8).

p) Where the selection of a new Aggregator by a port, as a result of changes to the selection parameters, results in other ports in the System being required to re-select their Aggregators in turn, this is achieved by setting Selected to UNSELECTED for those other ports that are required to re-select their Aggregators.

NOTE—The value of Selected is set to UNSELECTED by the Receive machine for the port when a change of LAG ID is detected.

q) A port shall not be enabled for use by the MAC Client until it has both selected and attached to an Aggregator.

5.4.14.2 Selection Logic—Recommended default operation

The recommended default behavior provides an element of determinism (i.e., history independence) in the assignment of ports to Aggregators. It also has the characteristic that no additional MAC addresses are needed, over and above those already assigned to the set of underlying MACs.

NOTE—This standard does not specify any alternative selection rules beyond the recommended set. A wide variety of selection rules are possible within the scope of the requirements stated in 5.4.14.1. In particular, it is possible within these requirements to support implementations that provide fewer Aggregators than ports, as well as implementations designed to minimize configuration changes at the expense of less deterministic behavior.

Each port has an Aggregator associated with it, (i.e., the number of Aggregators in the System equals the number of ports supported). Each port/Aggregator pair is assigned the same operational Key and port number. When there are multiple ports in an aggregation, the Aggregator that the set of ports selects is the Aggregator with the same port number as the lowest-numbered port in the aggregation. Note that this lowest numbered port may not be in a state that allows data transfer across the link; however, it has selected the Aggregator in question. This is illustrated in Figure 5–13.

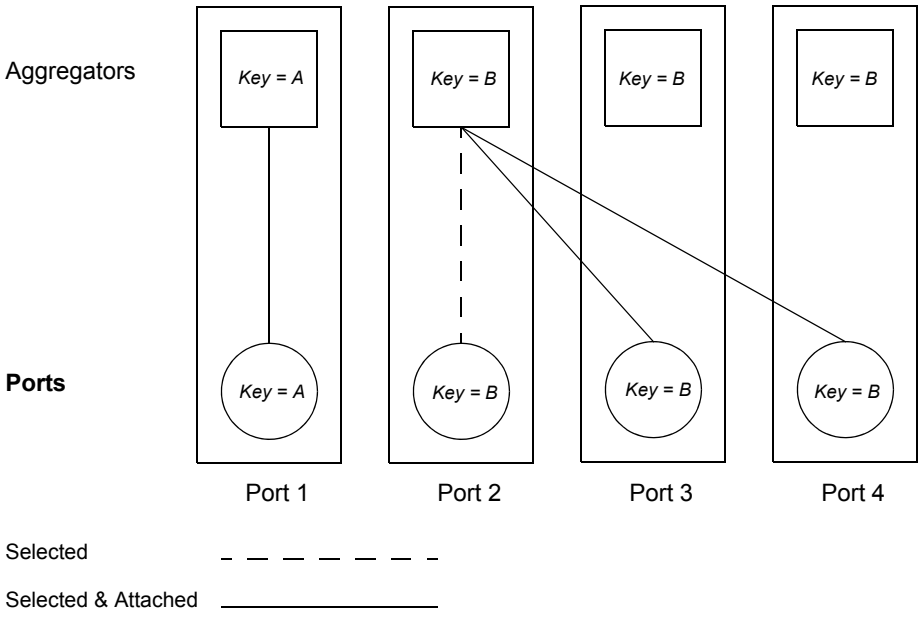


Figure 5–13—Selection of Aggregators

If the port is Individual, then the Aggregator selected is always the port's own Aggregator. Otherwise, an Aggregator is selected from the set of Aggregators corresponding to the set of ports that will form the aggregation. The Aggregator selected is the lowest numbered Aggregator with the same selection parameters as those of the port. These selection parameters are

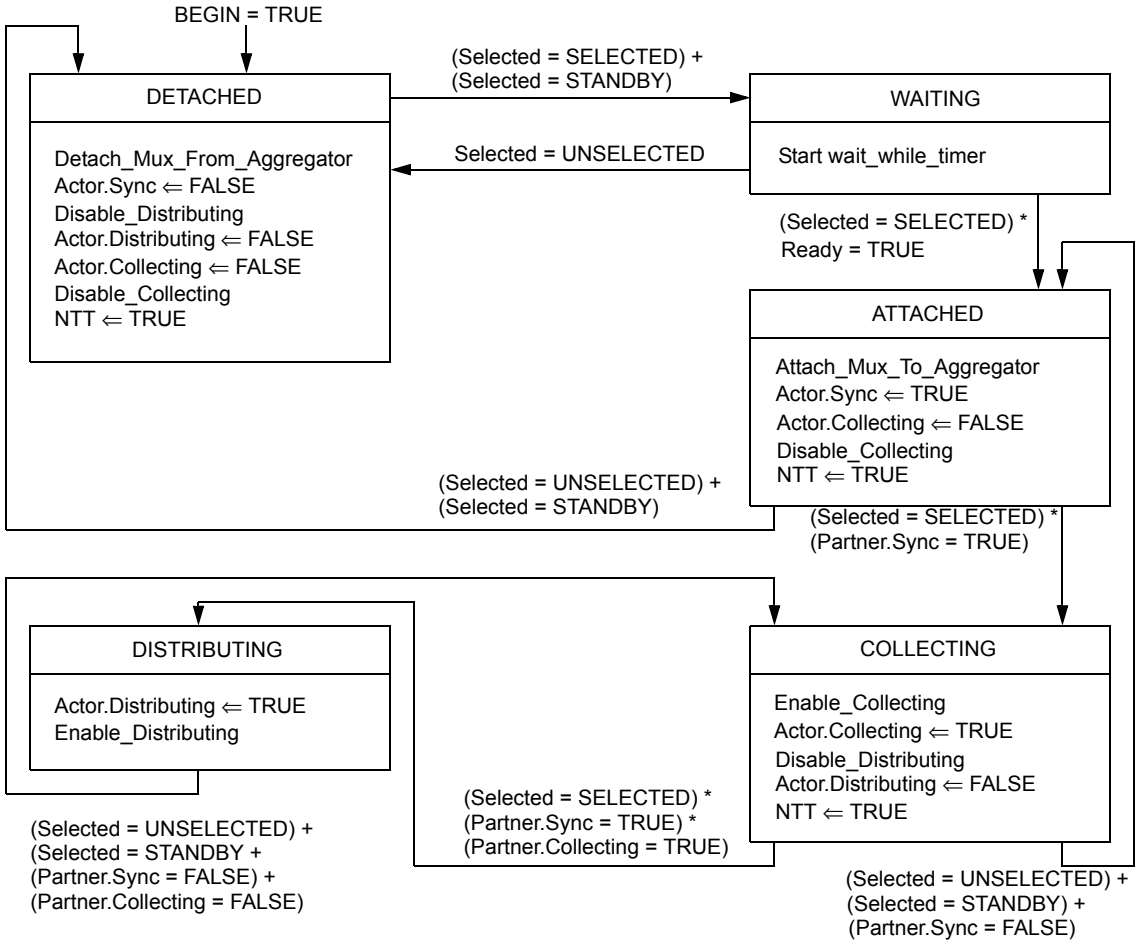
a) The Actor's System ID
b) The Actor's operational Key
c) The Partner's System ID
d) The Partner's operational Key
e) The Individual_Aggregator state (which must be FALSE)

5.4.15 Mux machine

The Mux machine shall implement the function specified in either of the Mux machine state diagrams, Figure 5–14 and Figure 5–15, with their associated parameters (5.4.4 through 5.4.11).

The state machine conventions in 5.1.1 assert that all in-state actions are instantaneous, are performed in sequence, and are performed prior to evaluating any exit conditions. While the Mux machine will operate correctly if all actions can be performed instantaneously, this will not be realistic in many implementations. Correct operation is maintained even if actions are not completed instantaneously, as long as each action completes prior to initiating the next sequential action, and all actions complete prior to evaluating any exit conditions.

The independent control state diagram (Figure 5–14) is suitable for use implementations in which it is possible to control enabling and disabling of frame collection from a port, and frame distribution to a port, independently. The coupled control state diagram (Figure 5–15) is suitable for use implementations where collecting and distributing cannot be controlled independently with respect to a port. It is recommended that the independent control state diagram be implemented in preference to the coupled control state diagram.



The following abbreviations are used in this diagram:
Actor.Sync: Actor_Oper_Port_State.Synchronization
Actor.Collecting: Actor_Oper_Port_State.Collecting
Actor.Distributing: Actor_Oper_Port_State.Distributing
Partner.Sync: Partner_Oper_Port_State.Synchronization
Partner.Collecting: Partner_Oper_Port_State.Collecting

Figure 5–14—Mux machine state diagram (independent control)

The value of the Selected variable may be changed by the following:

a) *The Receive machine.* The Receive machine can set Selected to UNSELECTED at any time if any of the following change: The Partner System ID, the Partner Priority, the Partner Key, the Partner_State.Aggregation, the Actor System ID, the Actor Priority, the Actor Key, or the Actor_State.Aggregation.

b) *The Selection Logic, in the process of selecting an Aggregator.* The Selection Logic will select an Aggregator when the Mux machine is in the DETACHED state and the value of the Selected variable is UNSELECTED.

c) *The Selection Logic, in the process of selecting or de-selecting standby links.* If the value of the Selected variable is SELECTED or STANDBY, the Selection Logic can change the value to STANDBY or SELECTED.

The Mux machine enters the WAITING state from the DETACHED state if the Selection Logic determines that Selected is either SELECTED or STANDBY. The WAITING state provides a holding state for the following two purposes:

d) If Selected is SELECTED, the wait_while_timer forces a delay to allow for the possibility that other ports may be reconfiguring at the same time. Once the wait_while_timer expires, and once the wait_while_timers of all other ports that are ready to attach to the same Aggregator have expired, the process of attaching the port to the Aggregator can proceed, and the state machine enters the ATTACHED state. During the waiting time, changes in selection parameters can occur that will result in a re-evaluation of Selected. If Selected becomes UNSELECTED, then the state machine re-enters the DETACHED state. If Selected becomes STANDBY, the operation is as described in item e).

   NOTE—This waiting period reduces the disturbance that will be visible to higher layers; for example, on start-up events. However, the selection need not wait for the entire waiting period in cases where it is known that no other ports will attach; for example, where all other ports with the same operational Key are already attached to the Aggregator.

e) If Selected is STANDBY, the port is held in the WAITING state until such a time as the selection parameters change, resulting in a re-evaluation of the Selected variable. If Selected becomes UNSELECTED, the state machine re-enters the DETACHED state. If SELECTED becomes SELECTED, then the operation is as described in item d). The latter case allows a port to be brought into operation from STANDBY with minimum delay once Selected becomes SELECTED.

On entry to the ATTACHED state, the Mux machine initiates the process of attaching the port to the selected Aggregator. Once the attachment process has completed, the value of Actor_Oper_Port_State.Synchronization is set to TRUE indicating that the Actor considers the port to be IN_SYNC, and Actor_Oper_Port_State.Collecting is set to FALSE. Collection of frames from the port is disabled. In the coupled control state machine, Distribution of frames to the port is also disabled, and Actor_Oper_Port_State.Distributing is set to FALSE.

A change in the Selected variable to UNSELECTED or to STANDBY causes the state machine to enter the DETACHED state. The process of detaching the port from the Aggregator is started. Once the detachment process is completed, Actor_Oper_Port_State.Synchronization is set to FALSE indicating that the Actor considers the port to be OUT_OF_SYNC, distribution of frames to the port is disabled, Actor_Oper_Port_State.Distributing and Actor_Oper_Port_State.Collecting are set to FALSE, and collection of frames from the port is disabled. The state machine remains in the DISABLED state until such time as the Selection logic is able to select an appropriate Aggregator.

While in the ATTACHED state, a TRUE value for Partner_Oper_Port_State.Synchronization causes the state machine to transition to the COLLECTING state (independent control) or the COLLECTING_DISTRIBUTING state (coupled control).

In the COLLECTING state, collection of frames from the port is enabled, then Actor_Oper_Port_State.Collecting is set to TRUE, then distribution of frames to the port is disabled and Actor_Port_Oper_State.Distributing is set to FALSE. The state machine will return to the ATTACHED state if Selected changes to UNSELECTED or STANDBY, or if the Partner's synchronization state becomes FALSE. Once the Partner has signaled that collecting has been enabled (Partner_Oper_Port_State.Collecting is TRUE), the state machine transitions to the DISTRIBUTING state. The value of Actor_Oper_Port_State.Distributing is set to TRUE, and then distribution of frames to the port is enabled. From DISTRIBUTING, a return to the COLLECTING state occurs if the value of Selected becomes UNSELECTED or STANDBY, if the Partner's synchronization state becomes FALSE, or if the Partner signals that collection has been disabled (Partner_Oper_Port_State.Collecting is FALSE).

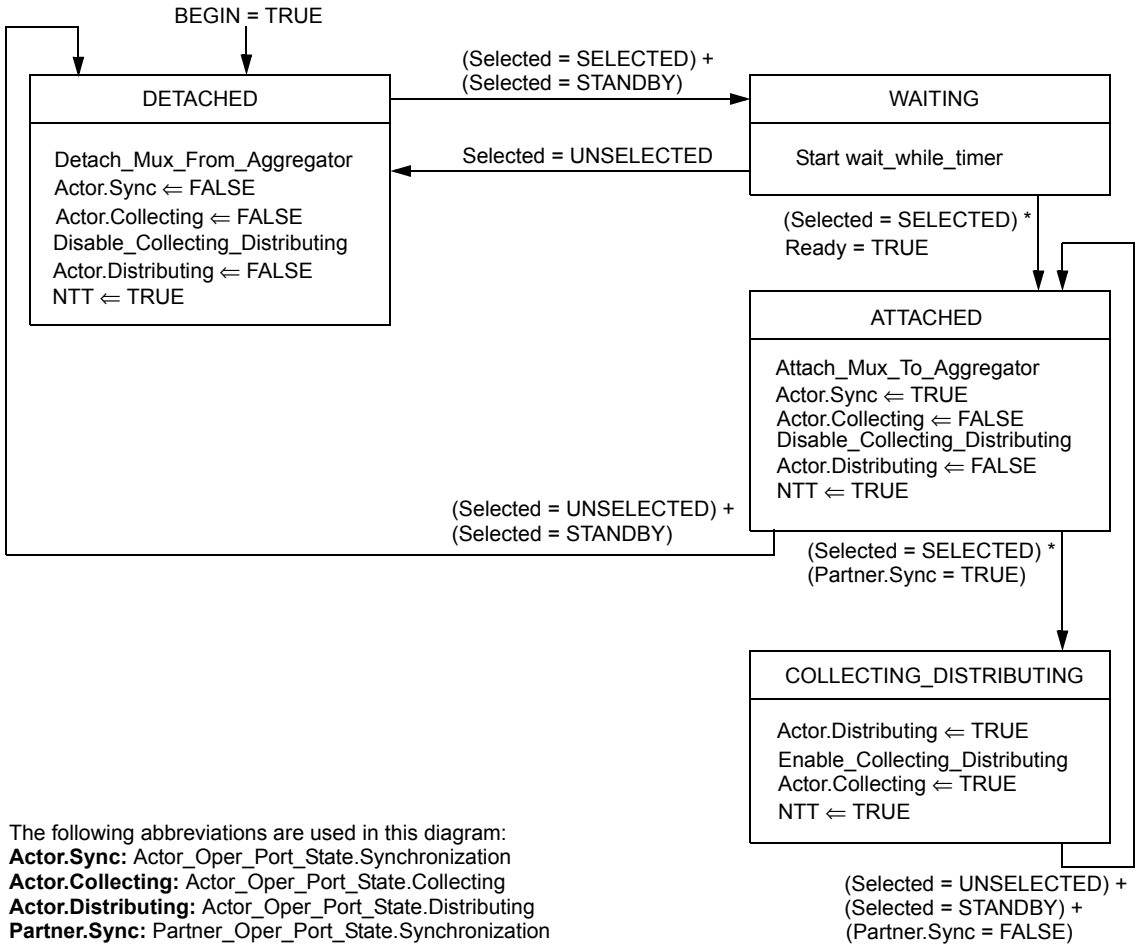


Figure 5–15—Mux machine state diagram (coupled control)

In the COLLECTING_DISTRIBUTING state, the value of Actor_Oper_Port_State.Distributing is set to TRUE, distribution of frames to the port and collection of frames from the port are both enabled, and then Actor_Oper_Port_State.Collecting is set to TRUE. The state machine will return to the ATTACHED state if Selected changes to UNSELECTED or STANDBY, or if the Partner's synchronization state becomes FALSE.

The sequence of operations and transitions defined for the COLLECTING and DISTRIBUTING states in the independent control version of this state machine ensures that frames are not distributed to a port until the Partner has enabled collection, and that distribution is stopped as soon as the Partner's state indicates that collection has been disabled. This sequence minimizes the possibility that frames will be misdelivered during the process of bringing the port into operation or taking the port out of operation. In the coupled control version of the state machine, the COLLECTING and DISTRIBUTING states merge together to form the combined state, COLLECTING_DISTRIBUTING. As independent control is not possible, the coupled control state machine does not wait for the Partner to signal that collection has started before enabling both collection and distribution.

The NTT variable is set to TRUE in the DETACHED, ATTACHED, COLLECTING, and COLLECTING_DISTRIBUTING states in order to ensure that the Partner is made aware of the changes in the Actor's state variables that are caused by the operations performed in those states.

5.4.16 Transmit machine

When the Transmit machine creates a LACPDU for transmission, it shall fill in the following fields with the corresponding operational values for this port:

a) Actor_Port and Actor_Port_Priority
b) Actor_System and Actor_System_Priority
c) Actor_Key
d) Actor_State
e) Partner_Port and Partner_Port_Priority
f) Partner_System and Partner_System_Priority
g) Partner_Key
h) Partner_State
i) CollectorMaxDelay

When the Periodic machine is in the NO_PERIODIC state, the Transmit machine shall

— Not transmit any LACPDUs, and
— Set the value of NTT to FALSE.

When the LACP_Enabled variable is TRUE and the NTT (5.4.7) variable is TRUE, the Transmit machine shall ensure that a properly formatted LACPDU (5.4.2) is transmitted [i.e., issue a CtrlMuxN:MA_DATA.Request(LACPDU) service primitive], subject to the restriction that no more than three LACPDUs may be transmitted in any Fast_Periodic_Time interval. If NTT is set to TRUE when this limit is in force, the transmission shall be delayed until such a time as the restriction is no longer in force. The NTT variable shall be set to FALSE when the Transmit machine has transmitted a LACPDU.

If the transmission of a LACPDU is delayed due to the above restriction, the information sent in the LACPDU corresponds to the operational values for the port at the time of transmission, not at the time when NTT was first set to TRUE. In other words, the LACPDU transmission model is based upon the transmission of state information that is current at the time an opportunity to transmit occurs, as opposed to queuing messages for transmission.

When the LACP_Enabled variable is FALSE, the Transmit machine shall not transmit any LACPDUs and shall set the value of NTT to FALSE.

5.4.17 Churn Detection machines

If implemented, the Churn Detection machines shall implement the functions specified in Figure 5–16 and Figure 5–17 with their associated parameters (5.4.4 through 5.4.11). Implementation of the Churn Detection machines is mandatory if the associated management functionality (the Aggregation Port Debug Information package) is implemented; otherwise, implementation of the Churn Detection machines is optional.

The Churn Detection machines detect the situation where a port is operable, but the Actor and Partner have not attached the link to an Aggregator and brought the link into operation within a bounded time period. Under normal operation of the LACP, agreement between Actor and Partner should be reached very rapidly. Continued failure to reach agreement can be symptomatic of device failure, of the presence of non-standard devices, or of misconfiguration; it can also be the result of normal operation in cases where either or both Systems are restricted in their ability to aggregate. Detection of this condition is signaled by the Churn Detection machines to management in order to prompt administrative action to further diagnose and correct the fault.

NOTE—One of the classes of problems that will be detected by this machine is the one where the implementation has been designed to support a limited number of Aggregators (fewer than the number of ports—see 5.6.4.2) and the physical topology is such that one or more ports end up with no Aggregator to attach to. This may be the result of a wiring error or an error in the allocation of operational Key values to the ports and Aggregators. Alternatively, failure of a link to aggregate may be the result of a link being placed in standby mode by a System that has hardware limitations placed on its aggregation ability, leading it to make use of the techniques described in 5.6 to find the ideal configuration. Given that the time taken by an aggregation-constrained System to stabilize its configuration may be relatively large, the churn detection timers allow 60 seconds to elapse before a Churn condition is signaled.

The symptoms that the Actor Churn Detection state machine detects is that the Actor's Mux has determined that it is OUT_OF_SYNC, and that condition has not resolved itself within a period of time equal to Short_Timeout_Time (5.4.4). Under normal conditions, this is ample time for convergence to take place. Similarly, the Partner Churn Detection state machine detects a failure of the Partner's Mux to synchronize.

The Actor Churn Detection state machine is depicted in Figure 5–16. The Partner Churn Detection state machine is depicted in Figure 5–17.

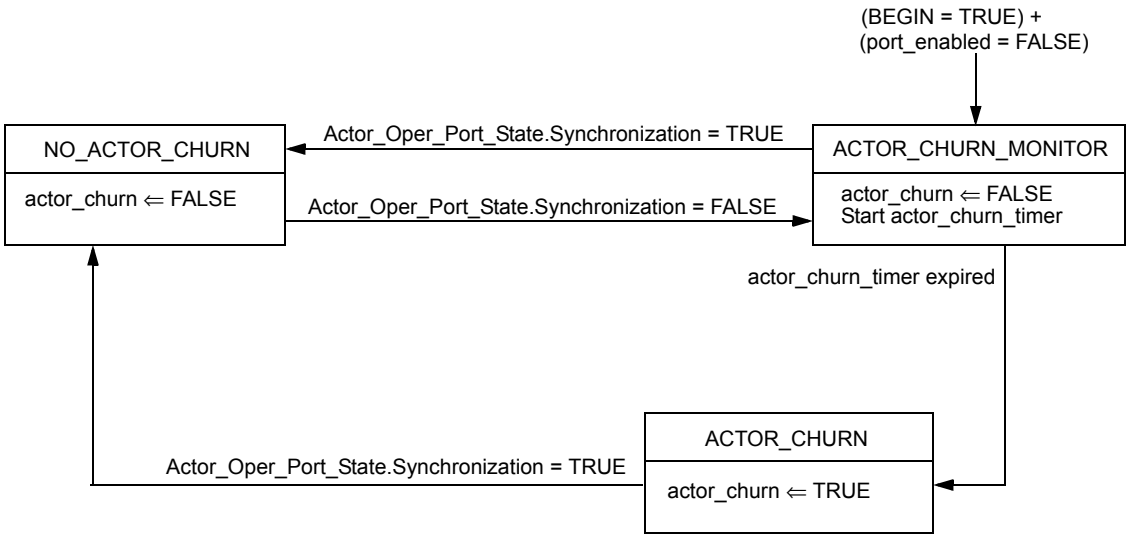


Figure 5–16—Actor Churn Detection machine state diagram

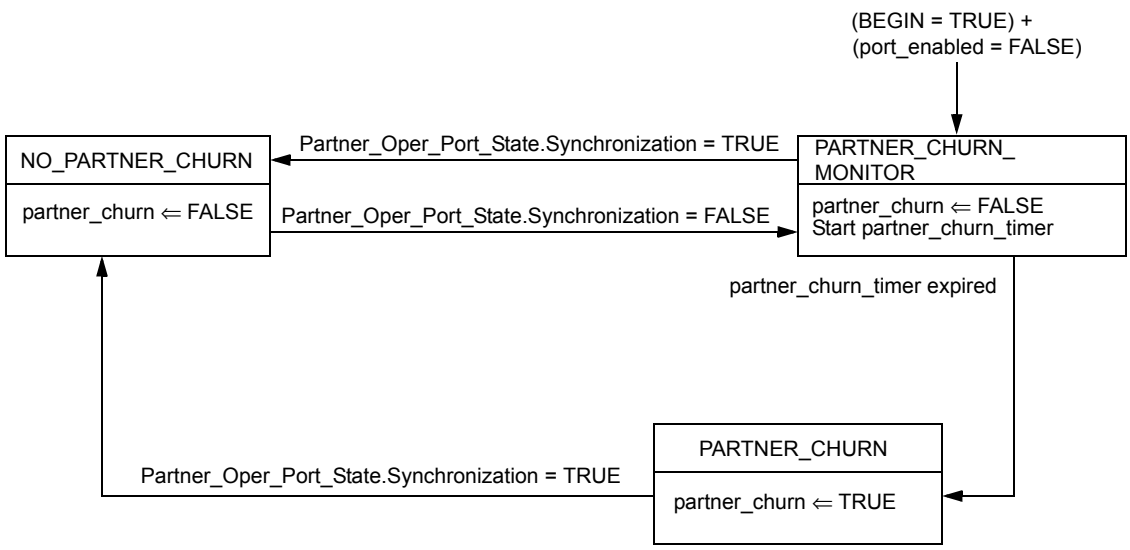


Figure 5–17—Partner Churn Detection machine state diagram

5.5 Marker protocol

5.5.1 Introduction

The Marker protocol allows the distribution function of an Actor's Link Aggregation sublayer to request the transmission of a Marker PDU on a given link. The Marker PDU is received by the Partner's collection function and a Marker Response PDU is returned on the same link to the initiating Actor's distribution function. Marker and Marker Response PDUs are treated by the underlying MACs at each end of the link as normal MAC Client PDUs; i.e., there is no prioritization or special treatment of Marker or Marker Response PDUs relative to other frames. Marker/Marker Response PDUs are subject to the operation of flow control, where supported on the link. Hence, if the distribution function requests transmission of a Marker PDU on a given link and does not transmit any further MAC Client PDUs that relate to a given set of conversations until the corresponding Marker Response PDU is received from that link, then it can be certain that there are no MAC Client PDUs related to those conversations still to be received by the Partner's collection function. The use of the Marker protocol can therefore allow the Distribution function a means of determining the point at which a given set of conversations can safely be reallocated from one link to another without the danger of causing frames in those conversations to be misordered at the Collector.

NOTE—The use of the Marker protocol is further discussed in Annex A.

The operation of the Marker protocol is unaffected by any changes in the Collecting and Distributing states associated with the port. Therefore, Marker and Marker Response PDUs can be sent on a port whose distribution function is disabled; similarly, such PDUs can be received and passed to the relevant Aggregator's collection or distribution function on a port whose collection function is disabled.

The use of the Marker protocol is optional; however, the ability to respond to Marker PDUs, as defined for the operation of the Marker Responder (see 5.5.4.1 and 5.5.4.2), is mandatory. Some distribution algorithms may not require the use of a marker; other mechanisms (such as timeouts) may be used as an alternative. As the specification of distribution algorithms is outside the scope of this standard, no attempt is made to specify how, when, or if the Marker protocol is used. (See Annex A for an informative discussion of distribution algorithms.)

The Marker protocol does not provide a guarantee of a response from the Partner; no provision is made for the consequences of frame loss or for the failure of the Partner System to respond correctly. Implementations that make use of this protocol must therefore make their own provision for handling such cases.

5.5.2 Sequence of operations

Figure 5–18 illustrates the sequence of marker operations between an initiating and responding System. Time is assumed to flow from the top of the diagram to the bottom.

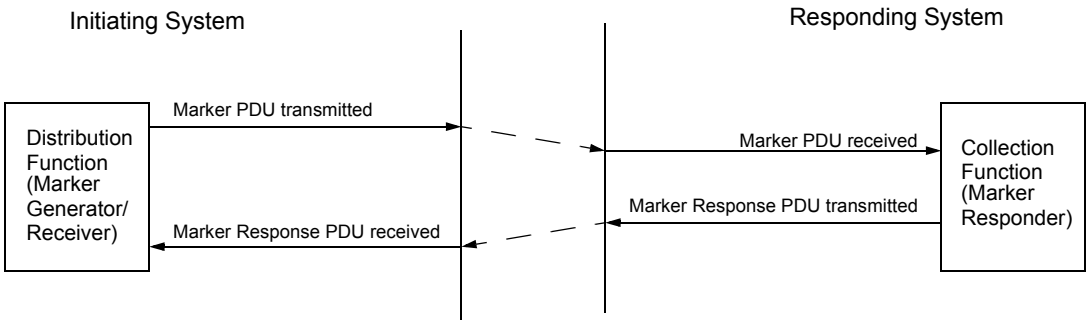


Figure 5–18—Marker protocol time sequence diagram

5.5.3 Marker and Marker Response PDU structure and encoding

5.5.3.1 Transmission and representation of octets

All Marker and Marker Response PDUs comprise an integral number of octets. The bits in each octet are numbered from 0 to 7, where 0 is the low-order bit. When consecutive octets are used to represent a numerical value, the most significant octet is transmitted first, followed by successively less significant octets.

When the encoding of (an element of) a Marker or Marker Response PDU is depicted in a diagram, then

a) Octets are transmitted from top to bottom.
b) Within an octet, bits are shown with bit 0 to the left and bit 7 to the right, and are transmitted from left to right.
c) Numerical values are encoded as binary numbers.

5.5.3.2 Marker and Marker Response PDU structure

The Marker PDU and Marker Response PDU structure shall be as shown in Figure 5–19 and as further described in the following field definitions:

a) *Destination Address.* The DA in all Marker and Marker Response PDUs is the Slow_Protocols_Multicast address. Its use and encoding are specified in IEEE Std 802.3 Annex 57A.
b) *Source Address.* The SA in Marker and Marker Response PDUs is the individual MAC address associated with the port from which the PDU is transmitted.
c) *Length/Type.* Marker and Marker Response PDUs are always Type encoded, and carry the Slow Protocol type field. The use and encoding of this type field is specified in IEEE Std 802.3 Annex 57A.
d) *Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. Both Marker and Marker Response PDUs carry the Marker_subtype value 0x02.

e) *Version number.* This identifies the Marker protocol version; implementations conformant to this version of the standard carry the value 0x01.

f) *TLV_type = Marker Information/Marker Response Information.* This indicates the nature of the information carried in this TLV-tuple. Marker Information is encoded as the integer value 0x01; Marker Response Information is encoded as the integer value 0x02.

g) *Marker_Information_Length/Marker_Response_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Both Marker and Marker Response information use a length value of 16 (0x10).

h) *Requester_Port.* The port number assigned to the port by the Requester (the System sending the initial Marker PDU), encoded as an unsigned integer.

Marker PDU	Octets	Marker Response PDU
Destination Address	6	Destination Address
Source Address	6	Source Address
Length/Type	2	Length/Type
Subtype = Marker	1	Subtype = Marker
Version Number	1	Version Number
TLV_type = Marker Information	1	TLV_type = Marker Response Information
Marker_Information_Length= 16	1	Marker_Response_Information_Length = 16
Requester_Port	2	Requester_Port
Requester_System	6	Requester_System
Requester_Transaction_ID	4	Requester_Transaction_ID
Pad = 0	2	Pad = 0
TLV_type = Terminator	1	TLV_type = Terminator
Terminator_Length = 0	1	Terminator_Length = 0
Reserved	90	Reserved
FCS	4	FCS

Figure 5–19—Marker PDU and Marker Response PDU structure

i) *Requester_System.* The Requester's System ID, encoded as a MAC address.

j) *Requester_Transaction_ID*. The transaction ID allocated to this request by the requester, encoded as an integer.

k) *Pad.* This field is used to align TLV-tuples on 16-byte memory boundaries. It is transmitted as zeros in Marker PDUs; in Marker Response PDUs, this field may be transmitted as zeros, or with the contents of this field from the Marker PDU triggering the response. The field is ignored on receipt in all cases.

NOTE—The difference in handling of the Pad field in Marker Response PDUs allows an implementation to reflect the contents of the received Marker PDU in its response, without enforcing the requirement to transmit the field as zeros.

l) *TLV_type = Terminator.* This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information carries the integer value 0x00.

m) *Terminator_Length.* This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

n) *Reserved*. These 90 octets are reserved for use in future extensions to the protocol. It is transmitted as zeros in Marker PDUs; in Marker Response PDUs, this field may be either transmitted as zeros, or with the contents of this field from the Marker PDU triggering the response. The field is ignored on receipt in all cases.

NOTE—These trailing reserved octets are included in all Marker and Marker Response PDUs in order to force a fixed PDU size, regardless of the version of the protocol. Hence, a Version 1 implementation is guaranteed to be able to receive version N PDUs successfully, although version N PDUs may contain additional information that cannot be interpreted (and will be ignored) by the Version 1 implementation. A crucial factor in ensuring backwards compatibility is that any future version of the protocol is required not to redefine the structure or semantics of information defined for the previous version; it may only add new information elements to the previous set. Hence, in a version N PDU, a Version 1 implementation can expect to find the Version 1 information in exactly the same places as in a Version 1 PDU, and can expect to interpret that information as defined for Version 1.

o) *FCS.* This field is the Frame Check Sequence, typically generated by the underlying MAC.

5.5.4 Protocol definition

5.5.4.1 Operation of the marker protocol

Marker PDUs may be generated by the Frame Distribution function to provide a sequence marker in the stream of frames constituting a conversation or set of conversations. Received Marker PDUs are delivered to the Marker Responder within the Frame Collection function of the Partner System.

On receipt of a valid Marker PDU, the Frame Collection function issues a Marker Response PDU, in the format specified in Figure 5–19, to the same port from which the Marker PDU was received. The **Requester_Port, Requester_System,** and **Requester_Transaction_ID** parameter in the Marker Response PDU carry the same values as those received in the corresponding Marker PDU.

Received Marker Response PDUs are passed to the Marker Receiver within the Frame Distribution function. Implementation of the Marker Generator and Receiver is optional.

The Marker Generator, if implemented, shall comply with the frame rate limitation constraint for Slow Protocols, as specified in IEEE Std 802.3 Annex 57A.3. A Marker Responder may (but is not required to) control its Marker Response transmissions to conform to this Slow Protocols timing constraint when faced with Marker messages not in compliance with this constraint (i.e., to send fewer Marker Response PDUs than Marker PDUs received). If the Marker Responder is controlling its responses in this manner, Marker Response PDUs corresponding to Marker PDUs received in excess of the Slow Protocols timing constraint shall not be sent.

NOTE—It is important that Marker Response PDUs not be queued indefinitely, and sent long after the corresponding Marker PDU that triggered the response.

Frames generated by the Marker Responder do not count towards the rate limitation constraint for Slow Protocols, as specified in IEEE Std 802.3 Annex 57A.3.

5.5.4.2 Marker Responder state diagram

The Marker Responder shall implement the function specified in Figure 5–20, with its associated parameters (5.5.4.2.1 through 5.5.4.2.3).

5.5.4.2.1 Constants

Slow_Protocols_Multicast

The value of the Slow Protocols reserved multicast address. (See IEEE Std 802.3 Table 57A–1.)

5.5.4.2.2 Variables

DA

SA

mac_service_data_unit

status

The parameters of the MA_DATA.request and MA_DATA.indication primitives, as defined in IEEE Std 802.3 Clause 2.

5.5.4.2.3 Messages

AggMuxN:MA_DATA.request

The service primitive used to transmit a frame with the specified parameters.

AggMuxN:MA_DATA.indication

The service primitive used to pass a received frame to a client with the specified parameters.

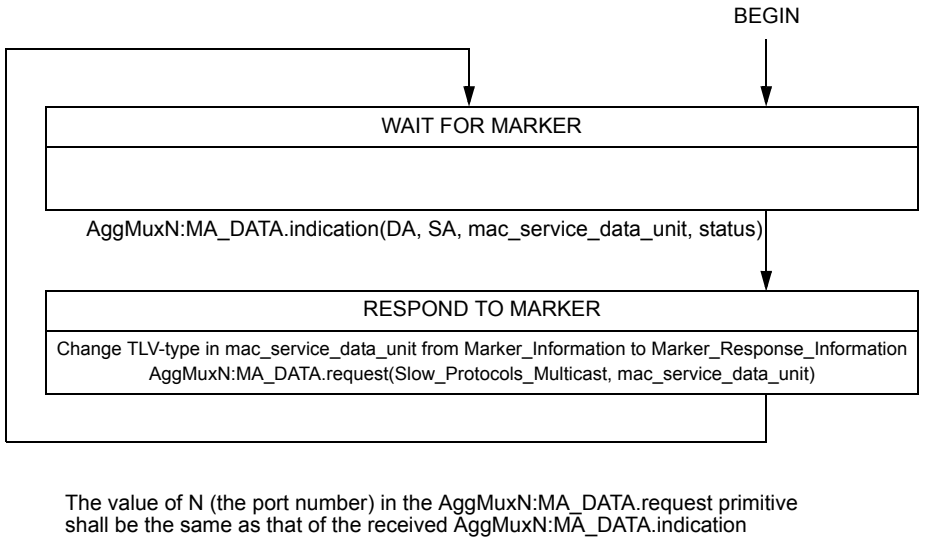


Figure 5–20—Marker Responder state diagram

Upon receipt of an AggMuxN:MA_DATA.indication primitive, the Marker Responder shall not validate the Version Number, Pad, or Reserved fields in the contained Marker Request PDU. The same actions are taken regardless of the values received in these fields. A Marker Responder may validate the Marker_Information_Length field. These behaviors, together with the constraint on future protocol enhancements discussed in the Note in 5.5.3.2 item n), allow Version 1 devices to be compatible with future revisions of the protocol.

5.6 Configuration capabilities and restrictions

5.6.1 Use of system and port priorities

The formulation chosen for the Link Aggregation Group identifier (5.3.6) has the consequence that it is not possible to represent two or more Link Aggregation Groups, comprising aggregateable links, that share the same combination of {SK, TL}. Hence, placing configuration restrictions on the size of an aggregation (e.g., for a Key Group containing six members, restricting the size of any aggregation to four members or fewer)

is only possible if it is also acceptable that only one Link Aggregation Group can be constructed from that Key Group for a given {SK, TL}. In practice, this restriction can be somewhat alleviated by subdividing Key Groups and allocating different operational Keys to each subdivision; however, this is, in general, only useful if the form of the size restriction is closely bound to physical subdivisions in the implementation (e.g., it might be possible to aggregate only those links that are on the same interface card).

In Systems that have limited aggregation capability of this form, the following algorithm shall be used to determine the subset of ports that will be aggregated together:

a) The System Aggregation Priority of each System is an eight octet binary number, formed by using the Actor_System_Priority as the two most significant octets and the Actor's System ID as the least significant six octets. For a given Actor and Partner, the System with the numerically lower value of System Aggregation Priority has the higher priority.

b) The Port Aggregation Priority of each port is a four octet binary number, formed by using the Actor_Port_Priority as the two most significant octets and the port number as the two least significant octets. For any given set of ports, the port with the numerically lower value of Port Aggregation Priority has the higher priority.

c) Ports shall be selected for aggregation by each System based upon the Port Aggregation Priority assigned by the System with the higher System Aggregation Priority, starting with the highest priority port of the System with the higher priority, and working downward through the ordered list of Port Aggregation Priority values for the N ports, applying the particular constraints imposed on the System concerned.

d) For each link that a given System cannot include in the aggregation, the Selection Logic identifies the Selection state of the corresponding port as STANDBY, preventing the link from becoming active. The synchronization state signaled in transmitted LACPDUs for such links will be OUT_OF_SYNC.

e) The selection algorithm is reapplied upon changes in the membership of the Link Aggregation Group (for example, if a link fails, or if a new link joins the group) and any consequent changes to the set of active links are made accordingly.

A port that is selected as standby as a result of limitations on aggregation capability can be viewed as providing a "hot standby" facility, as it will be able to take part in the aggregation upon failure of one of the active links in the aggregation. The ability to hold links in a standby mode in this way provides the possibility of using LACP even where the System is incapable of supporting distribution and collection with more than one port. Parallel links could be automatically configured as standby links, and deployed to mask link failures without any disruption to higher layer protocols.

5.6.2 Dynamic allocation of operational Keys

In some circumstances, the use of System and port priorities may prove to be insufficient to generate the optimum aggregation among the set of links connecting a pair of Systems. A System may have a limited aggregation capability that cannot be simply expressed as a limit on the total number of links in the aggregation. The full description of its restrictions may be that it can only aggregate together particular subsets of links, and the sizes of the subsets need not all be the same.

NOTE 1—An example would be an implementation organized such that, for a set of four links A through D, it would be possible to operate with {A+B+C+D} as a single aggregation, or operate with {A+B} and {C+D} as two separate aggregations, or operate as four individual links; however, all other aggregation possibilities (such as {A+C} and {B+D}) would not be achievable by the implementation.

In such circumstances, it is permissible for the System with the higher System Aggregation Priority (i.e., the numerically lower value) to dynamically modify the operational Key value associated with one or more of the ports; the System with the lower priority shall not attempt to modify operational Key values for this purpose. Operational Key changes made by the higher priority System should be consistent with maintaining its highest priority port in the aggregate as an active link (i.e., in the IN_SYNC state).

Successive operational Key changes, if they occur, should progressively reduce the number of ports in the aggregation. The original operational Key value should be maintained for the highest priority port thought to be aggregateable.

NOTE 2—Restricting operational Key changes in the manner described prevents the case where both Partner Systems involved have limited capability and both attempt to make operational Key changes; this could be a non-converging process, as a change by one participant can cause the other participant to make a change, which in turn causes the first participant to make a change—and so on, ad infinitum.

This approach effectively gives the higher priority System permission to search the set of possible configurations, in order to find the best combination of links given its own and its Partner's configuration constraints. The reaction of the Partner System to these changes can be determined by observing the changes in the synchronization state of each link. A System performing operational Key changes should allow at least 4 s for the Partner System to change an OUT_OF_SYNC state to an IN_SYNC state.

In the course of normal operation a port can dynamically change its operating characteristics (e.g., data rate, full or half duplex operation). It is permissible (and appropriate) for the operational Key value associated with such a port to change with the corresponding changes in the operating characteristics of the link, so that the operational Key value always correctly reflects the aggregation capability of the link. Operational Key changes that reflect such dynamic changes in the operating characteristics of a link may be made by either System without restriction.

5.6.3 Link Aggregation on shared-medium links

The Link Aggregation Control Protocol cannot detect the presence of multiple Aggregation-aware devices on the same link. Hence, shared-medium links shall be treated as Individual, with transmission/reception of LACPDUs disabled on such ports.

5.6.4 Selection Logic variants

Two variants of the Selection Logic rules are described as follows:

a) The first accommodates implementations that may wish to operate in a manner that minimizes disturbance of existing aggregates, at the expense of the deterministic characteristics of the logic described in 5.4.14.2.

b) The second accommodates implementations that may wish to limit the number of Aggregators that are available for use to fewer than the number of ports supported.

5.6.4.1 Reduced reconfiguration

By removing the constraint that the Aggregator chosen is always the lowest numbered Aggregator associated with the set of ports in an aggregation, an implementation can minimize the degree to which changes in the membership of a given aggregation result in changes of connectivity at higher layers. As there would still be the same number of Aggregators and ports with a given operational Key value, any port will still always be able to find an appropriate Aggregator to attach to, however the configuration achieved over time (i.e., after a series of link disconnections, reconnections, or reconfigurations) with this relaxed set of rules would not necessarily be the same as the configuration achieved if all Systems involved were reset, given the rules stated in 5.4.15.

5.6.4.2 Limited Aggregator availability

By removing the constraint that there are always as many Aggregators as ports, an implementation can limit the number of MAC Client interfaces available to higher layers while maintaining the ability for each Aggregator to serve multiple ports. This has the same effect as removing the assumption that Aggregators and their associated ports have the same operational Key value; Aggregators can be effectively disabled (and

therefore ignored) by configuring their Keys to be different from any operational Key value allocated to any of the ports.

In this scenario, any port(s) that cannot find a suitable Aggregator to attach to will simply wait in the DETACHED state until an Aggregator becomes available, with a synchronization state of OUT_OF_SYNC.

5.7 Protocol implementation conformance statement (PICS) proforma for Clause 5, Aggregation of Multiple Link Segments[7]

5.7.1 Introduction

The supplier of an implementation that is claimed to conform to Clause 5, Link Aggregation, 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. A PICS is included at the end of each clause as appropriate. The PICS can be used for a variety of purposes by various parties, including the following:

a) As a checklist by the protocol implementer, to reduce the risk of failure to conform to the standard through oversight;

b) As a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma, by the supplier and acquirer, or potential acquirer, of the implementation;

c) As a basis for initially checking the possibility of interworking with another implementation by the user, or potential user, of the implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICS);

d) As the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation, by a protocol tester.

5.7.2 Abbreviations and special symbols

The following symbols are used in the PICS proforma:

M	mandatory field/function
!	negation
O	optional field/function
O.<n>	optional field/function, but at least one of the group of options labeled by the same numeral <n> is required
O/<n>	optional field/function, but one and only one of the group of options labeled by the same numeral <n> is required
X	prohibited field/function
<item>:	simple-predicate condition, dependent on the support marked for <item>
<item1>*<item2>:	AND-predicate condition, the requirement must be met if both optional items are implemented

5.7.3 Instructions for completing 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 subclauses, each containing a group of items. Answers to the questionnaire items are to be provided in the right-most column, either by simply marking an answer to indicate a restricted choice (usually Yes, No, or Not Applicable), 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.)

[7] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this subclause so that it can be used for its intended purpose and may further publish the completed PICS.

Each item is identified by an item reference in the first column; the second column contains the question to be answered; the third column contains the reference or references to the material that specifies the item in the main body of the standard; the sixth column contains values and/or comments pertaining to the question to be answered. The remaining columns record the status of the items—whether the support is mandatory, optional or conditional—and provide the space for the answers.

The 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 or presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the protocol implementation conformance statement for the implementation in question.

Note that where an implementation is capable of being configured in more than one way, according to the items listed under Major Capabilities/Options, 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, if that would make presentation of the information easier and clearer.

5.7.4 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 the 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 a brief rationale, based perhaps upon specific application needs, for the exclusion of features that, although optional, are nonetheless commonly present in implementations.

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.

5.7.5 Exceptional information

It may occasionally happen that a supplier will wish to answer an item with mandatory or prohibited 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 is required to write into the Support column an X<i> reference to an item of Exception Information, and to 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 that a possible reason for the situation described above is that a defect in the standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

5.7.6 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, optional, or prohibited—are dependent upon whether or not certain other items are supported.

Individual conditional items are indicated by a conditional symbol of the form "<item>:<s>" in the Status column, where "<item>" is an item reference that appears in the first column of the table for some other item, and "<s>" is a status symbol, M (Mandatory), O (Optional), or X (Not Applicable).

If the item referred to by the conditional symbol is marked as supported, then 1) the conditional item is applicable, 2) its status is given by "<s>", and 3) the support column is to be completed in the usual way. Otherwise, the conditional item is not relevant and the Not Applicable (N/A) answer is to be marked.

Each item whose reference is used in a conditional symbol is indicated by an asterisk in the Item column.

5.7.7 Identification

5.7.7.1 Implementation identification

Supplier (Note 1)	
Contact point for queries about the PICS (Note 1)	
Implementation Name(s) and Version(s) (Notes 1 and 3)	
Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names (Note 2)	
NOTE 1—Required for all implementations. NOTE 2—May be completed as appropriate in meeting the requirements for the identification. NOTE 3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).	

5.7.7.2 Protocol summary

Identification of protocol specification	IEEE Std 802.1AX-2008, Clause 5, Link Aggregation.
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	

5.7.8 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
*MG	Marker Generator/Receiver	5.2.5		O	Yes [ ] No [ ]
*AM	Aggregation Port Debug Information package support	6.3		O	Yes [ ] No [ ]
*CM	Churn Detection machine	5.4.17	Required if Aggregation Port Debug Information package supported	AM: M !AM: O	N/A [ ] M:Yes [ ] Yes [ ] No [ ]

5.7.9 Frame Collector

Item	Feature	Subclause	Value/Comment	Status	Support
FC1	Frame Collector function	5.2.3	As specified in the state diagram shown in Figure 5–4 and associated definitions in 5.2.3.1	M	Yes [ ]
FC2	Frame Collector function—CollectorMaxDelay	5.2.3.1.4	Deliver or discard frames within CollectorMaxDelay	M	Yes [ ]

5.7.10 Frame Distributor

Item	Feature	Subclause	Value/Comment	Status	Support
	Distribution algorithm ensures the following, when frames received by Frame Collector:	5.2.4			
FD1	Frame misordering		None	M	Yes [ ]
FD2	Frame duplication		None	M	Yes [ ]
FD3	Frame Distributor function	5.2.4	Function as specified in the state diagram shown in Figure 5–5 and associated definitions in 5.2.4.1	M	Yes [ ]

5.7.11 Marker protocol

Item	Feature	Subclause	Value/Comment	Status	Support
MGR1	Marker Generator/Receiver	5.2.5		MG:M	N/A [ ] M:Yes [ ]
MGR2	Marker Responder	5.2.6	Function specified in 5.5.4.2	M	Yes [ ]

5.7.12 Aggregator Parser/Multiplexer

Item	Feature	Subclause	Value/Comment	Status	Support
APM1	Aggregator Multiplexer	5.2.7	Transparent pass-through of frames	M	Yes [ ]
APM2	Aggregator Multiplexer	5.2.7	Discard of TX frames when port not Distributing	M	Yes [ ]
APM3	Aggregator Parser	5.2.7	Function specified by state diagram shown in Figure 5–6 and associated definitions in 5.2.7.1	M	Yes [ ]
APM4	Aggregator Parser	5.2.7	Discard of RX frames when port not Collecting	M	Yes [ ]

5.7.13 Control Parser/Multiplexer

Item	Feature	Subclause	Value/Comment	Status	Support
CPM1	Control Multiplexer	5.2.9	Transparent pass-through of frames	M	Yes [ ]
CPM2	Control Parser	5.2.9	Function specified by state diagram shown in Figure 5–7 and associated definitions in 5.2.9.1	M	Yes [ ]

5.7.14 System identification

Item	Feature	Subclause	Value/Comment	Status	Support
SID1	Globally unique identifier	5.3.2	Globally administered individual MAC address plus System Priority	M	Yes [ ]
SID2	MAC address chosen	5.3.2	MAC address associated with one of the ports	O	Yes [ ] No [ ]

5.7.15 Aggregator identification

Item	Feature	Subclause	Value/Comment	Status	Support
AID1	Globally unique identifier	5.3.3	Globally administered individual MAC address	M	Yes [ ]
AID2	Integer identifier	5.3.3	Uniquely identifies the Aggregator within the System	M	Yes [ ]
*AID3	Unique identifier allocated	5.3.3	Unique identifier assigned to one of its bound ports	O	Yes [ ] No [ ]
AID4			Unique identifier not assigned to any other Aggregator	!AID3 :M	N/A [ ] M:Yes [ ]

5.7.16 Port identification

Item	Feature	Subclause	Value/Comment	Status	Support
PID1	Port identifiers	5.3.4	Unique within a System; port number 0 not used for any port	M	Yes [ ]

5.7.17 Capability identification

Item	Feature	Subclause	Value/Comment	Status	Support
CID1	Administrative and operational Key values associated with each port	5.3.5		M	Yes [ ]
CID2	Administrative and operational Key values associated with each Aggregator	5.3.5		M	Yes [ ]

5.7.18 Link Aggregation Group identification

Item	Feature	Subclause	Value/Comment	Status	Support
LAG1	LAG ID component values	5.3.6.1	Actor's values non-zero. Partner's admin values only zero for Individual ports	M	Yes [ ]

5.7.19 Detaching a link from an Aggregator

Item	Feature	Subclause	Value/Comment	Status	Support
DLA1	Effect on conversation reallocated to a different link	5.3.13	Frame ordering preserved	M	Yes [ ]

5.7.20 LACPDU structure

Item	Feature	Subclause	Value/Comment	Status	Support
LPS2	LACPDU structure	5.4.2.2	As shown in Figure 5–8 and as described	M	Yes [ ]
LPS3	LACPDU structure	5.4.2	All Reserved octets ignored on receipt and transmitted as zero	M	Yes [ ]

5.7.21 State machine variables

Item	Feature	Subclause	Value/Comment	Status	Support
SMV1	Partner_Admin_Port_State	5.4.7	Collecting set to the same value as Synchronization	M	Yes [ ]
SMV2	LACP_Enabled	5.4.8	FALSE for half duplex ports, otherwise TRUE	M	Yes [ ]

5.7.22 Receive machine

Item	Feature	Subclause	Value/Comment	Status	Support
RM1	Receive machine	5.4.12	As defined in Figure 5–11 and associated parameters	M	Yes [ ]
RM2	Validation of LACPDUs		No validation of Version Number, TLV_type, or Reserved fields	M	Yes [ ]

5.7.23 Periodic Transmission machine

Item	Feature	Subclause	Value/Comment	Status	Support
PM1	Periodic Transmission machine	5.4.13	As defined in Figure 5–12 and associated parameters	M	Yes [ ]

5.7.24 Selection Logic

Item	Feature	Subclause	Value/Comment	Status	Support
	Selection logic requirements	5.4.14.1			
SLM1	Aggregator support		At least one Aggregator per System	M	Yes [ ]
SLM2	Port Keys		Each port assigned an operational Key	M	Yes [ ]
SLM3	Aggregator Keys		Each Aggregator assigned an operational Key	M	Yes [ ]
SLM4	Aggregator Identifiers		Each Aggregator assigned an identifier	M	Yes [ ]
SLM5	Aggregator selection		If same Key assignment as port	M	Yes [ ]
SLM6	Ports that are members of the same Link Aggregation Group		Ports select same Aggregator	M	Yes [ ]
SLM7	Pair of ports connected in loop-back		Not select same Aggregator as each other	M	Yes [ ]
SLM8	Port required to be Individual		Not select same Aggregator as any other port	M	Yes [ ]
SLM9	Port is Aggregateable		Not select same Aggregator as any Individual port	M	Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
SLM10	Port unable to select an Aggregator		Port not attached to any Aggregator	M	Yes [ ]
SLM11	Further aggregation constraints		Ports may be selected as standby	O	Yes [ ] No [ ]
SLM12	Selected variable		Set to SELECTED or STANDBY once Aggregator is determined	M	Yes [ ]
SLM13	Port enabled		Only when selected and attached to an Aggregator	M	Yes [ ]
SLM14	Recommended default operation of Selection Logic	5.4.14.2	Meets requirements of 5.4.14.2	O	Yes [ ] No [ ]

5.7.25 Mux machine

Item	Feature	Subclause	Value/Comment	Status	Support
XM1	Mux machine	5.4.15	As defined in Figure 5–14 or Figure 5–15, and associated parameters	M	Yes [ ]

5.7.26 Transmit machine

Item	Feature	Subclause	Value/Comment	Status	Support
	Transmitted in outgoing LACP-DUs	5.4.16			
TM1	Actor_Port and Actor_Port_Priority	5.4.16		M	Yes [ ]
TM2	Actor_System and Actor_System_Priority	5.4.16		M	Yes [ ]
TM3	Actor_Key	5.4.16		M	Yes [ ]
TM4	Actor_State	5.4.16		M	Yes [ ]
TM5	Partner_Port and Partner_Port_Priority	5.4.16		M	Yes [ ]
TM6	Partner_System and Partner_System_Priority	5.4.16		M	Yes [ ]
TM7	Partner_Key	5.4.16		M	Yes [ ]
TM8	Partner_State	5.4.16		M	Yes [ ]
TM9	CollectorMaxDelay	5.4.16		M	Yes [ ]
TM10	Action when Periodic machine is in the NO_PERIODIC state	5.4.16	Set NTT to FALSE, do not transmit	M	Yes [ ]
TM11	Action when LACP_Enabled is TRUE, NTT is TRUE, and not rate limited	5.4.16	Properly formatted LACPDU transmitted	M	Yes [ ]
TM12	Action when LACP_Enabled is TRUE and NTT is TRUE, when rate limit is in force	5.4.16	Transmission delayed until limit is no longer in force	M	Yes [ ]
TM13	Action when LACPDU has been transmitted	5.4.16	Set NTT to FALSE	M	Yes [ ]
TM14	Action when LACP_Enabled is FALSE	5.4.16	Set NTT to FALSE, do not transmit	M	Yes [ ]

5.7.27 Churn Detection machines

Item	Feature	Subclause	Value/Comment	Status	Support
CM1	Churn Detection machines	5.4.17	As defined in Figure 5–16 and Figure 5–17	CM:M	N/A [ ] Yes [ ]

5.7.28 Marker protocol

Item	Feature	Subclause	Value/Comment	Status	Support
FP1	Respond to all received Marker PDUs	5.5.1	As specified by 5.5.4	M	Yes [ ]
FP2	Use of the Marker protocol	5.5.1	As specified by 5.5.4	O	Yes [ ] No [ ]
FP3	MARKER.request service primitives request rate	5.5.4.1	Maximum of five during any one-second period	MG:M	N/A [ ] Yes [ ]
FP6	Marker PDU structure	5.5.3.2	As shown in Figure 5–19 and as described	MG:M	N/A [ ] Yes [ ]
FP7	Marker Response PDU structure	5.5.3.2	As shown in Figure 5–19 and as described	M	Yes [ ]
FP8	Marker Responder State Diagram	5.5.4.2	As specified in Figure 5–20 and 5.5.4.2.1 through 5.5.4.2.3	M	Yes [ ]
FP9	Validation of Marker Request PDUs	5.5.4.2.3	Marker Responder shall not validate the Version Number, Pad, or Reserved fields	M	Yes [ ]

5.7.29 Configuration capabilities and restrictions

Item	Feature	Subclause	Value/Comment	Status	Support
CCR1	Algorithm used to determine subset of ports that will be aggregated in Systems that have limited aggregation capability	5.6.1	As specified in items a) to e) of 5.6.1	M	Yes[ ]
CCR2	Key value modification to generate optimum aggregation	5.6.2		O	Yes [ ] No [ ]
CCR3	Key value modification when System has higher System Aggregation Priority			CCR2:M	N/A [ ] M:Yes [ ]
CCR4	Key value modification when System has lower System Aggregation Priority			CCR2:X	N/A [ ] X:Yes [ ]

5.7.30 Link Aggregation on shared-medium links

Item	Feature	Subclause	Value/Comment	Status	Support
LSM1	Shared-medium links—Configuration	5.6.3	Configured as Individual links	M	Yes [ ]
LSM2	Shared-medium links—Operation of LACP	5.6.3	LACP is disabled	M	Yes [ ]

6. Management

6.1 Overview

This clause provides the layer management specification for Link Aggregation. It provides the objects, attributes, and behaviours to support Link Aggregation.

The layout of this clause takes the same form as IEEE Std 802.3 Clause 30. It identifies a common management model and framework applicable to the IEEE 802.1AX managed elements, identifies those elements and defines their managed objects, attributes, and behaviours in a protocol-independent language.

It defines facilities comprised of a set of statistics and actions needed to provide IEEE 802.1AX management services. The Procedural Model provides a formal description of the relationship between Link Aggregation and the layer management facilities.

This management specification has been developed in accordance with the OSI management architecture as specified in the ISO Management Framework document, ISO/IEC 7498-4: 1989. It is independent of any particular management application or management protocol.

The management facilities defined in this standard may be accessed both locally and remotely. Thus, the layer management specification provides facilities that can be accessed from within a station or can be accessed remotely by means of a peer-management protocol operating between application entities.

In this standard, no peer management facilities are necessary for initiating or terminating normal protocol operations or for handling abnormal protocol conditions. Since these activities are subsumed by the normal operation of the protocol, they are not considered to be a function of layer management and are, therefore, not discussed in this clause.

Implementation of part or all of layer management is not a requirement for conformance to any other clause of this standard.

The improper use of some of the facilities described in this clause may cause serious disruption of the network. In accordance with ISO management architecture, any necessary security provisions should be provided by the agent in the Local System Environment. This can be in the form of specific security features or in the form of security features provided by the peer communication facilities.

6.1.1 Systems management overview

Within the ISO Open Systems Interconnection (OSI) architecture, the need to handle the special problems of initializing, terminating, and monitoring ongoing activities and assisting in their operations, as well as handling abnormal conditions, is recognized. These needs are collectively addressed by the systems management component of the OSI architecture.

A management protocol is required for the exchange of information between systems on a network. This management standard is independent of any particular management protocol.

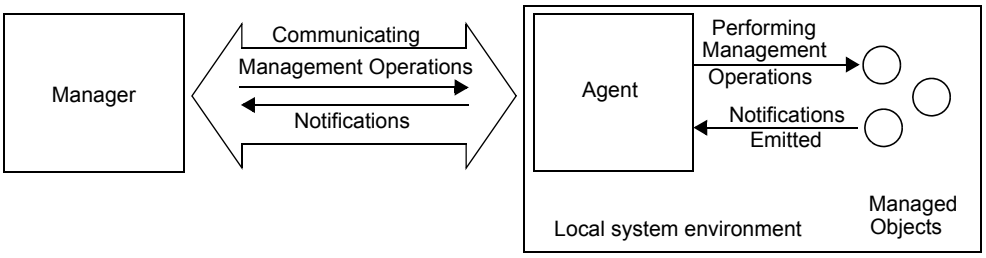
This management standard, in conjunction with the management standards of other layers, provides the means to perform various management functions. IEEE 802.1AX management collects information needed from, and provides a means to exercise control over, Link Aggregation.

The relationship between the various management entities and the layer entities according to the ISO model is shown in Figure 6–1.

6.1.2 Management model

This standard describes management of Link Aggregation in terms of a general model of management of resources within the open systems environment. The model, which is described in ISO/IEC 10040: 1992, is briefly summarized here.

Management is viewed as a distributed application modeled as a set of interacting management processes. These processes are executed by systems within the open environment. A managing system executes a managing process that invokes management operations. A managed system executes a process that is receptive to these management operations and provides an interface to the resources to be managed. A managed object is the abstraction of a resource that represents its properties as seen by (and for the purpose of) management. Managed objects respond to a defined set of management operations. Managed objects are also capable of emitting a defined set of notifications. This interaction of processes is shown in Figure 6–1.



NOTE—This figure is drawn from Figure 1 of ISO/IEC 10040: 1992, Information technology—Open Systems Interconnection—Systems management overview. In the event of any conflict, the depiction in ISO/IEC 10040: 1992 takes precedence.

Figure 6–1—Interaction between manager, agent, and objects

A managed object is a management view of a resource. The resource may be a logical construct, function, physical device, or anything subject to management. Managed objects are defined in terms of four types of elements:

a) *Attributes.* Data-like properties (as seen by management) of a managed object.
b) *Actions.* Operations that a managing process may perform on an object or its attributes.
c) *Notifications.* Unsolicited reports of events that may be generated by an object.
d) *Behaviour.* The way in which managed objects, attributes, and actions interact with the actual resources they model and with each other.

The above items are defined in 6.3 of this clause in terms of the template requirements of ISO/IEC 10165-4: 1992.

Some of the functions and resources within IEEE 802.1AX devices are appropriate targets for management. They have been identified by specifying managed objects that provide a management view of the functions or resources. Within this general model, the IEEE 802.1AX device is viewed as a managed device. It performs functions as defined by the applicable standard for such a device. Managed objects providing a view of those functions and resources appropriate to the management of the device are specified. The purpose of this standard is to define the object classes associated with the devices in terms of their attributes, operations, notifications, and behaviour.

6.2 Managed objects

6.2.1 Introduction

This clause identifies the Managed Object classes for IEEE 802.1AX components within a managed system. It also identifies which managed objects and packages are applicable to which components.

All counters defined in this specification are assumed to be wrap-around counters. Wrap-around counters are those that automatically go from their maximum value (or final value) to zero and continue to operate. These unsigned counters do not provide for any explicit means to return them to their minimum (zero), i.e., reset. Because of their nature, wrap-around counters should be read frequently enough to avoid loss of information. When a counter has a maximum increment rate specified at one speed of operation, and that counter is appropriate to a higher speed of operation, then the maximum increment rate at that higher speed of operation is as shown in Equation (6–1).

$$\text{maximum increment rate specified} \times \left(\frac{\text{higher speed of operation in Mb/s}}{\text{specified speed of operation in Mb/s}} \right) \quad (6–1)$$

unless otherwise indicated.

6.2.2 Overview of managed objects

Managed objects provide a means to

- Identify a resource
- Control a resource
- Monitor a resource

6.2.2.1 Text description of managed objects

In case of conflict, the formal behaviour definitions in 6.3 take precedence over the text descriptions in this subclause.

oAggPortDebugInformation
: A single instance of oAggPortDebugInformation may be contained within oAggregationPort. This managed object class provides optional additional information that can assist with debugging and fault finding in Systems that support Link Aggregation.

oAggPortStats
: A single instance of oAggPortStats may be contained within oAggregationPort. This managed object class provides optional additional statistics related to LACP and Marker protocol activity on an instance of an Aggregation Port that is involved in Link Aggregation.

oAggregationPort
: oAggregationPort is contained within oAggregator. An instance of this managed object class is present for each Aggregation Port that is part of the aggregation represented by the oAggregator instance. This managed object class provides the basic management controls necessary to allow an instance of an Aggregation Port to be managed, for the purposes of Link Aggregation.

oAggregator
: oAggregator is the top-most managed object class of the Link Aggregation containment tree shown in Figure 6–2. Note that this managed object class may be contained within another superior managed

object class. Such containment is expected, but is outside the scope of this International Standard. The oAggregator managed object class provides the management controls necessary to allow an instance of an Aggregator to be managed.

6.2.3 Containment

A containment relationship is a structuring relationship for managed objects in which the existence of a managed object is dependent on the existence of a containing managed object. The contained managed object is said to be the subordinate managed object, and the containing managed object the superior managed object. The containment relationship is used for naming managed objects. The local containment relationships among object classes are depicted in the entity relationship diagram Figure 6–2. The figure show the names of the object classes and whether a particular containment relationship is one-to-one or one-to-many. For further requirements on this topic, see IEEE Std 802.1F™-1993.

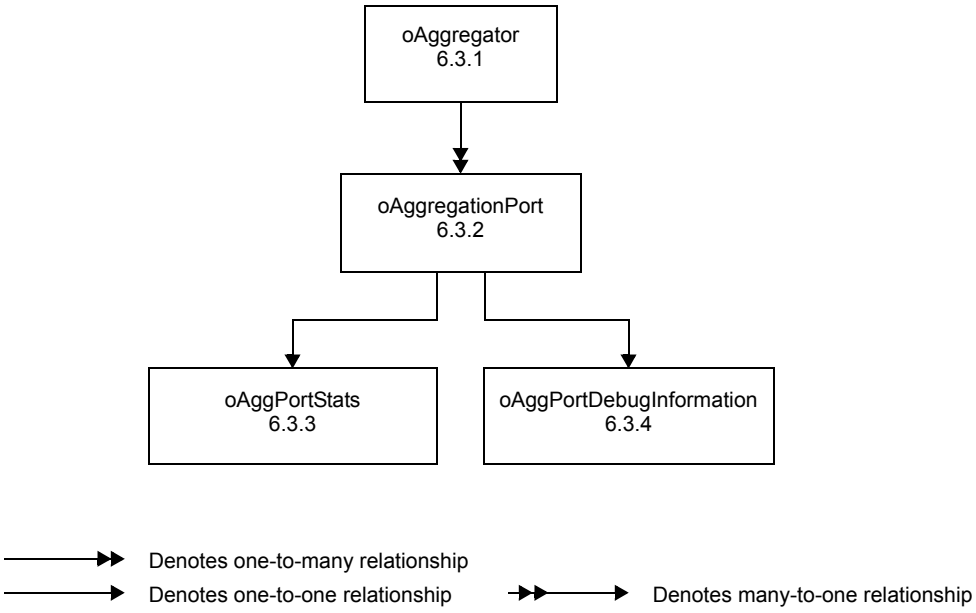


Figure 6–2—Link aggregation entity relationship diagram

6.2.4 Naming

The name of an individual managed object is hierarchically defined within a managed system. For example a Aggregator port might be identified as “aggregator 3, port 13” that is, port 13 of aggregator 3 within the managed system.

6.2.5 Capabilities

This standard makes use of the concept of *packages* as defined in ISO/IEC 10165-4: 1992 as a means of grouping behaviour, attributes, actions, and notifications within a managed object class definition. Packages may either be mandatory, or be conditional, that is to say, present if a given condition is true. Within this standard *capabilities* are defined, each of which corresponds to a set of packages, which are components of a number of managed object class definitions and which share the same condition for presence. Implementation of the Basic and Mandatory packages is the minimum requirement for claiming conformance to IEEE 802.1AX Management. Implementation of an entire optional capability is required in

order to claim conformance to that capability. The capabilities and packages for IEEE 802.1AX Management are specified in Table 6–1.

Table 6–1—Link Aggregation capabilities

				DTE					
	Object name	Object type	Operations supported	Basic package (mandatory)	Mandatory package (mandatory)	Recommended package (optional)	Optional package (optional)	Aggregation Port statistics (optional)	Aggregation Port debug information (optional)
oAggregator (6.3.1)									
	aAggID	ATTRIBUTE	GET	X					
	aAggDescription	ATTRIBUTE	GET		X				
	aAggName	ATTRIBUTE	GET-SET		X				
	aAggActorSystemID	ATTRIBUTE	GET-SET		X				
	aAggActorSystemPriority	ATTRIBUTE	GET-SET		X				
	aAggAggregateOrIndividual	ATTRIBUTE	GET		X				
	aAggActorAdminKey	ATTRIBUTE	GET-SET		X				
	aAggActorOperKey	ATTRIBUTE	GET		X				
	aAggMACAddress	ATTRIBUTE	GET		X				
	aAggPartnerSystemID	ATTRIBUTE	GET		X				
	aAggPartnerSystemPriority	ATTRIBUTE	GET		X				
	aAggPartnerOperKey	ATTRIBUTE	GET		X				
	aAggAdminState	ATTRIBUTE	GET-SET		X				
	aAggOperState	ATTRIBUTE	GET		X				
	aAggTimeOfLastOperChange	ATTRIBUTE	GET		X				
	aAggDataRate	ATTRIBUTE	GET		X				
	aAggOctetsTxOK	ATTRIBUTE	GET			X			
	aAggOctetsRxOK	ATTRIBUTE	GET			X			
	aAggFramesTxOK	ATTRIBUTE	GET		X				
	aAggFramesRxOK	ATTRIBUTE	GET		X				
	aAggMulticastFramesTxOK	ATTRIBUTE	GET				X		
	aAggMulticastFramesRxOK	ATTRIBUTE	GET				X		
	aAggBroadcastFramesTxOK	ATTRIBUTE	GET				X		
	aAggBroadcastFramesRxOK	ATTRIBUTE	GET				X		
	aAggFramesDiscardedOnTx	ATTRIBUTE	GET			X			
	aAggFramesDiscardedOnRx	ATTRIBUTE	GET			X			
	aAggFramesWithTxErrors	ATTRIBUTE	GET			X			
	aAggFramesWithRxErrors	ATTRIBUTE	GET			X			
	aAggUnknownProtocolFrames	ATTRIBUTE	GET			X			
	aAggLinkUpDownNotificationEnable	ATTRIBUTE	GET-SET		X				

Table 6–1—Link Aggregation capabilities *(continued)*

				DTE					
	Object name	Object type	Operations supported	Basic package (mandatory)	Mandatory package (mandatory)	Recommended package (optional)	Optional package (optional)	Aggregation Port statistics (optional)	Aggregation Port debug information (optional)
	nAggLinkUpNotification	NOTIFICATION			X				
	nAggLinkDownNotification	NOTIFICATION			X				
	aAggPortList	ATTRIBUTE	GET			X			
	aAggCollectorMaxDelay	ATTRIBUTE	GET-SET		X				
oAggregationPort (6.3.2)									
	aAggPortID	ATTRIBUTE	GET	X					
	aAggPortActorSystemPriority	ATTRIBUTE	GET-SET		X				
	aAggPortActorSystemID	ATTRIBUTE	GET		X				
	aAggPortActorAdminKey	ATTRIBUTE	GET-SET		X				
	aAggPortActorOperKey	ATTRIBUTE	GET		X				
	aAggPortPartnerAdminSystemPriority	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerOperSystemPriority	ATTRIBUTE	GET		X				
	aAggPortPartnerAdminSystemID	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerOperSystemID	ATTRIBUTE	GET		X				
	aAggPortPartnerAdminKey	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerOperKey	ATTRIBUTE	GET		X				
	aAggPortSelectedAggID	ATTRIBUTE	GET		X				
	aAggPortAttachedAggID	ATTRIBUTE	GET		X				
	aAggPortActorPort	ATTRIBUTE	GET		X				
	aAggPortActorPortPriority	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerAdminPort	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerOperPort	ATTRIBUTE	GET		X				
	aAggPortPartnerAdminPortPriority	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerOperPortPriority	ATTRIBUTE	GET		X				
	aAggPortActorAdminState	ATTRIBUTE	GET-SET		X				
	aAggPortActorOperState	ATTRIBUTE	GET		X				
	aAggPortPartnerAdminState	ATTRIBUTE	GET-SET		X				
	aAggPortPartnerOperState	ATTRIBUTE	GET		X				
	aAggPortAggregateOrIndividual	ATTRIBUTE	GET		X				
oAggPortStats (6.3.3)									
	aAggPortStatsID	ATTRIBUTE	GET					X	
	aAggPortStatsLACPDUsRx	ATTRIBUTE	GET					X	
	aAggPortStatsMarkerPDUsRx	ATTRIBUTE	GET					X	

Table 6–1—Link Aggregation capabilities *(continued)*

				DTE					
	Object name	Object type	Operations supported	Basic package (mandatory)	Mandatory package (mandatory)	Recommended package (optional)	Optional package (optional)	Aggregation Port statistics (optional)	Aggregation Port debug information (optional)
	aAggPortStatsMarkerResponsePDUsRx	ATTRIBUTE	GET					X	
	aAggPortStatsUnknownRx	ATTRIBUTE	GET					X	
	aAggPortStatsIllegalRx	ATTRIBUTE	GET					X	
	aAggPortStatsLACPDUsTx	ATTRIBUTE	GET					X	
	aAggPortStatsMarkerPDUsTx	ATTRIBUTE	GET					X	
	aAggPortStatsMarkerResponsePDUsTx	ATTRIBUTE	GET					X	
oAggPortDebugInformation (6.3.4)									
	aAggPortDebugInformationID	ATTRIBUTE	GET						X
	aAggPortDebugRxState	ATTRIBUTE	GET						X
	aAggPortDebugLastRxTime	ATTRIBUTE	GET						X
	aAggPortDebugMuxState	ATTRIBUTE	GET						X
	aAggPortDebugMuxReason	ATTRIBUTE	GET						X
	aAggPortDebugActorChurnState	ATTRIBUTE	GET						X
	aAggPortDebugPartnerChurnState	ATTRIBUTE	GET						X
	aAggPortDebugActorChurnCount	ATTRIBUTE	GET						X
	aAggPortDebugPartnerChurnCount	ATTRIBUTE	GET						X
	aAggPortDebugActorSyncTransitionCount	ATTRIBUTE	GET						X
	aAggPortDebugPartnerSyncTransitionCount	ATTRIBUTE	GET						X
	aAggPortDebugActorChangeCount	ATTRIBUTE	GET						X
	aAggPortDebugPartnerChangeCount	ATTRIBUTE	GET						X
Common Attributes Template									
	aCMCounter	ATTRIBUTE	GET		X	X	X	X	X

6.3 Management for Link Aggregation

6.3.1 Aggregator managed object class

This subclause formally defines the behaviours for the oAggregator managed object class, attributes, and notifications.

Some of the attributes that are part of the definition of the oAggregator managed object class are derived by summing counter values from attributes of other objects; e.g., to generate a count of received frames for the Aggregator, the individual value for each Aggregation Port contributes to the sum. Where calculations of

this form are used, the values that contribute to the Aggregator's attributes are *increments* in the values of the component attributes, not their absolute values. As any individual Aggregation Port is potentially only temporarily attached to its current Aggregator, the count values it contributes to the Aggregator's counters are the increments in its values that it has experienced during the period of time that it has been attached to that Aggregator.

The counter values defined for the Aggregator have been formulated as far as possible to make the Aggregator behave like an individual IEEE 802.3 MAC. The counts of frames received and transmitted are formulated to reflect the counts that would be expected by the MAC Client; they do not include frames transmitted and received as part of the operation of LACP or the Marker protocol, only frames that pass through the interface between the MAC Client and the Aggregator. However, as LACP and the Marker protocol are, as far as the individual MACs are concerned, part of their MAC Client, the RX/TX counters for the individual MACs will reflect both control and data traffic. As counts of errors at the port level cannot always be cleanly delineated between those that occurred as a result of aggregation activity and those that did not, no attempt has been made to separate these aspects of the port error counts. Therefore, there is not necessarily a direct correspondence between the individual MAC counters and the corresponding derived counters at the Aggregator level.

It should also be noted that the counters defined for the Aggregator include values that can only apply to half duplex links. This is consistent with the approach taken in Link Aggregation that a link that can only operate as an individual link is nonetheless considered as being attached to an Aggregator. This simplifies the modelling of managed objects for links that can operate in either half or full duplex, and ensures a consistent presentation of the attributes regardless of the type of links attached to the Aggregator.

NOTE—The operation of Auto-Negotiation may mean that a given link can operate in full duplex or half duplex, depending upon the capabilities of the device(s) connected to it. Keeping the management view the same regardless of a link's current mode of operation allows a consistent management approach to be taken across all types of links.

6.3.1.1 Aggregator attributes

6.3.1.1.1 aAggID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The unique identifier allocated to this Aggregator by the local System. This attribute identifies an Aggregator instance among the subordinate managed objects of the containing object. This value is read-only.

NOTE—The aAggID is represented in the SNMP MIB as an ifIndex—see C.4.2.;

6.3.1.1.2 aAggDescription

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOUR DEFINED AS:

A human-readable text string containing information about the Aggregator. This string could include information about the distribution algorithm in use on this Aggregator; for example, "Aggregator 1, Dist Alg=Dest MAC address." This string is read-only. The contents are vendor specific.;

6.3.1.1.3 aAggName

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOUR DEFINED AS:

A human-readable text string containing a locally significant name for the Aggregator. This string is read-write.;

6.3.1.1.4 aAggActorSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:

MACAddress

BEHAVIOUR DEFINED AS:

A 6-octet read-write MAC address value used as a unique identifier for the System that contains this Aggregator.

NOTE—From the perspective of the Link Aggregation mechanisms described in Clause 5, only a single combination of Actor's System ID and System Priority are considered, and no distinction is made between the values of these parameters for an Aggregator and the port(s) that are associated with it (i.e., the protocol is described in terms of the operation of aggregation within a single System). However, the managed objects provided for the Aggregator and the port both allow management of these parameters. The result of this is to permit a single piece of equipment to be configured by management to contain more than one System from the point of view of the operation of Link Aggregation. This may be of particular use in the configuration of equipment that has limited aggregation capability (see 5.6).;

6.3.1.1.5 aAggActorSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

A 2-octet read-write value indicating the priority value associated with the Actor's System ID.;

6.3.1.1.6 aAggAggregateOrIndividual

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOUR DEFINED AS:

A read-only Boolean value indicating whether the Aggregator represents an Aggregate ("TRUE") or an Individual link ("FALSE").;

6.3.1.1.7 aAggActorAdminKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current administrative value of the Key for the Aggregator. The administrative Key value may differ from the operational Key value for the reasons discussed in 5.6.2. This is a 16-bit read-write value. The meaning of particular Key values is of local significance.;

6.3.1.1.8 aAggActorOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:
The current operational value of the Key for the Aggregator. The administrative Key value may differ from the operational Key value for the reasons discussed in 5.6.2. This is a 16-bit read-only value. The meaning of particular Key values is of local significance.;

6.3.1.1.9 aAggMACAddress

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOUR DEFINED AS:
A 6-octet read-only value carrying the individual MAC address assigned to the Aggregator.;

6.3.1.1.10 aAggPartnerSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOUR DEFINED AS:
A 6-octet read-only MAC address value consisting of the unique identifier for the current protocol Partner of this Aggregator. A value of zero indicates that there is no known Partner. If the aggregation is manually configured, this System ID value will be a value assigned by the local System.;

6.3.1.1.11 aAggPartnerSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:
A 2-octet read-only value that indicates the priority value associated with the Partner's System ID. If the aggregation is manually configured, this System Priority value will be a value assigned by the local System.;

6.3.1.1.12 aAggPartnerOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:

The current operational value of the Key for the Aggregator's current protocol Partner. This is a 16-bit read-only value. If the aggregation is manually configured, this Key value will be a value assigned by the local System.;

6.3.1.1.13 aAggAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
up
down

BEHAVIOUR DEFINED AS:

This read-write value defines the administrative state of the Aggregator. A value of "up" indicates that the operational state of the Aggregator (aAggOperState) is permitted to be either up or down. A value of "down" forces the operational state of the Aggregator to be down. Changes to the administrative state affect the operational state of the Aggregator only, not the operational state of the Aggregation Ports that are attached to the Aggregator. A GET operation returns the current administrative state. A SET operation changes the administrative state to a new value.;

6.3.1.1.14 aAggOperState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
up
down

BEHAVIOUR DEFINED AS:

This read-only value defines the operational state of the Aggregator. The operational state is "up" if one or more of the Aggregation Ports that are attached to the Aggregator are collecting, or both collecting and distributing, and if the value of aAggAdminState for the Aggregator is also "up." If none of the Aggregation Ports that are attached to the Aggregator are collecting and/or distributing, or if there are no Aggregation Ports attached to this Aggregator, then the operational state is "down." An operational state of "up" indicates that the Aggregator is available for use by the MAC Client; a value of "down" indicates that the Aggregator is not available for use by the MAC Client.;

6.3.1.1.15 aAggTimeOfLastOperChange

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The value of aTimeSinceSystemReset (See IEEE Std 802.3 Annex F.2.1) at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a value of zero. This value is read-only.;

6.3.1.1.16 aAggDataRate

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current data rate, in bits per second, of the aggregate link. The value is calculated as N times the data rate of a single link in the aggregation, where N is the number of active links. This attribute is read-only.;

6.3.1.1.17 aAggOctetsTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 1 230 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the data and padding octets transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation. The count does not include octets transmitted by the Aggregator in frames that carry LACPDUs or Marker PDUs (6.3.3.1.7, 6.3.3.1.8, 6.3.3.1.9). However, it includes frames discarded by the Distribution function of the Aggregator (6.3.1.1.25). This value is read-only.;

6.3.1.1.18 aAggOctetsRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 1 230 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the data and padding octets received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation. The count does not include octets received in frames that carry LACP or Marker PDUs (6.3.3.1.2, 6.3.3.1.3, 6.3.3.1.4), or frames discarded by the Collection function of the Aggregator (6.3.1.1.26). This value is read-only.;

6.3.1.1.19 aAggFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the data frames transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (6.3.3.1.7, 6.3.3.1.8, 6.3.3.1.9). However, it includes frames discarded by the Distribution function of the Aggregator (6.3.1.1.25). This value is read-only.;

6.3.1.1.20 aAggFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the data frames received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames that carry LACP or Marker PDUs (6.3.3.1.2, 6.3.3.1.3, 6.3.3.1.4), or frames discarded by the Collection function of the Aggregator (6.3.1.1.26). This value is read-only.;

6.3.1.1.21 aAggMulticastFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the data frames transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation, to a group destination address other than the broadcast address. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (6.3.3.1.7, 6.3.3.1.8, 6.3.3.1.9). However, it includes frames discarded by the Distribution function of the Aggregator (6.3.1.1.25). This value is read-only.;

6.3.1.1.22 aAggMulticastFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the data frames received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation, that were addressed to an active group address other than the broadcast address. The count does not include frames that carry LACP or Marker PDUs (6.3.3.1.2, 6.3.3.1.3, 6.3.3.1.4), or frames discarded by the Collection function of the Aggregator (6.3.1.1.26). This value is read-only.;

6.3.1.1.23 aAggBroadcastFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the broadcast data frames transmitted by this Aggregator on all Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (6.3.3.1.7, 6.3.3.1.8, 6.3.3.1.9). However, it includes frames discarded by the Distribution function of the Aggregator (6.3.1.1.25). This value is read-only.;

6.3.1.1.24 aAggBroadcastFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of the broadcast data frames received by this Aggregator, from the Aggregation Ports that are (or have been) members of the aggregation. The count does not include frames that carry LACP or Marker PDUs (6.3.3.1.2, 6.3.3.1.3, 6.3.3.1.4), illegal or unknown protocol frames (6.3.3.1.5, 6.3.3.1.6), or frames discarded by the Collection function of the Aggregator (6.3.1.1.26). This value is read-only.;

6.3.1.1.25 aAggFramesDiscardedOnTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of data frames requested to be transmitted by this Aggregator that were discarded by the Distribution function of the Aggregator when conversations are re-allocated to different ports, due to the requirement to ensure that the conversations are flushed on the old ports in order to maintain proper frame ordering (43A.3), or discarded as a result of excessive collisions by ports that are (or have been) members of the aggregation. This value is read-only.;

6.3.1.1.26 aAggFramesDiscardedOnRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of data frames, received on all ports that are (or have been) members of the aggregation, that were discarded by the Collection function of the Aggregator as they were received on ports whose Collection function was disabled. This value is read-only.;

6.3.1.1.27 aAggFramesWithTxErrors

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of data frames requested to be transmitted by this Aggregator that experienced transmission errors on ports that are (or have been) members of the aggregation. This count does not include frames discarded due to excess collisions. This value is read-only.;

6.3.1.1.28 aAggFramesWithRxErrors

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of data frames discarded on reception by all ports that are (or have been) members of the aggregation, or that were discarded by the Collection function of the Aggregator, or that

were discarded by the Aggregator due to the detection of an illegal Slow Protocols PDU (6.3.3.1.6). This value is read-only.;

6.3.1.1.29 aAggUnknownProtocolFrames

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 16 000 counts per second for a single 10 Mb/s aggregation.

BEHAVIOUR DEFINED AS:

A count of data frames discarded on reception by all ports that are (or have been) members of the aggregation, due to the detection of an unknown Slow Protocols PDU (6.3.3.1.5). This value is read-only.;

6.3.1.1.30 aAggPortList

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF INTEGERs that match the syntax of aAggPortID.

BEHAVIOUR DEFINED AS:

The value of this read-only attribute contains the list of Aggregation Ports that are currently attached to the Aggregator. An empty list indicates that there are no Aggregation Ports attached. Each integer value in the list carries an aAggPortID attribute value (6.3.2.1.1).;

6.3.1.1.31 aAggLinkUpDownNotificationEnable

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
enabled
disabled

BEHAVIOUR DEFINED AS:

When set to “enabled,” Link Up and Link Down notifications are enabled for this Aggregator. When set to “disabled,” Link Up and Link Down notifications are disabled for this Aggregator. This value is read-write.;

6.3.1.1.32 aAggCollectorMaxDelay

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The value of this 16-bit read-write attribute defines the maximum delay, in tens of microseconds, that may be imposed by the Frame Collector between receiving a frame from an Aggregator Parser, and either delivering the frame to its MAC Client or discarding the frame (see 5.2.3.1.1).;

6.3.1.2 Aggregator Notifications

6.3.1.2.1 nAggLinkUpNotification

NOTIFICATION

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:
When aAggLinkUpDownNotificationEnable is set to "enabled," a Link Up notification is generated when the Operational State of the Aggregator changes from "down" to "up." When aAggLinkUpDownNotificationEnable is set to "disabled," no Link Up notifications are generated. The notification carries the identifier of the Aggregator whose state has changed.;

6.3.1.2.2 nAggLinkDownNotification

NOTIFICATION

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:
When aAggLinkUpDownNotificationEnable is set to "enabled," a Link Down notification is generated when the Operational State of the Aggregator changes from "up" to "down." When aAggLinkUpDownNotificationEnable is set to "disabled," no Link Down notifications are generated. The notification carries the identifier of the Aggregator whose state has changed.;

6.3.2 Aggregation Port managed object class

This subclause formally defines the behaviours for the oAggregationPort managed object class attributes.

6.3.2.1 Aggregation Port Attributes

6.3.2.1.1 aAggPortID

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:
The unique identifier allocated to this Aggregation Port by the local System. This attribute identifies an Aggregation Port instance among the subordinate managed objects of the containing object. This value is read-only.

NOTE—The aAggPortID is represented in the SNMP MIB as an ifIndex—see C.4.4.;

6.3.2.1.2 aAggPortActorSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:
A 2-octet read-write value used to define the priority value associated with the Actor's System ID.;

6.3.2.1.3 aAggPortActorSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:
MACAddress

BEHAVIOUR DEFINED AS:

A 6-octet read-only MAC address value that defines the value of the System ID for the System that contains this Aggregation Port.;

6.3.2.1.4 aAggPortActorAdminKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current administrative value of the Key for the Aggregation Port. This is a 16-bit read-write value. The meaning of particular Key values is of local significance.;

6.3.2.1.5 aAggPortActorOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current operational value of the Key for the Aggregation Port. This is a 16-bit read-only value. The meaning of particular Key values is of local significance.;

6.3.2.1.6 aAggPortPartnerAdminSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

A 2-octet read-write value used to define the administrative value of priority associated with the Partner's System ID. The assigned value is used, along with the value of aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.;

6.3.2.1.7 aAggPortPartnerOperSystemPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

A 2-octet read-only value indicating the operational value of priority associated with the Partner's System ID. The value of this attribute may contain the manually configured value carried in aAggPortPartnerAdminSystemPriority if there is no protocol Partner.;

6.3.2.1.8 aAggPortPartnerAdminSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:

MACAddress

BEHAVIOUR DEFINED AS:

A 6-octet read-write MACAddress value representing the administrative value of the Aggregation Port's protocol Partner's System ID. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminKey, aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.;

6.3.2.1.9 aAggPortPartnerOperSystemID

ATTRIBUTE

APPROPRIATE SYNTAX:

MACAddress

BEHAVIOUR DEFINED AS:

A 6-octet read-only MACAddress value representing the current value of the Aggregation Port's protocol Partner's System ID. A value of zero indicates that there is no known protocol Partner. The value of this attribute may contain the manually configured value carried in aAggPortPartnerAdminSystemID if there is no protocol Partner.;

6.3.2.1.10 aAggPortPartnerAdminKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current administrative value of the Key for the protocol Partner. This is a 16-bit read-write value. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID, aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.;

6.3.2.1.11 aAggPortPartnerOperKey

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current operational value of the Key for the protocol Partner. The value of this attribute may contain the manually configured value carried in aAggPortPartnerAdminKey if there is no protocol Partner. This is a 16-bit read-only value.;

6.3.2.1.12 aAggPortSelectedAggID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The identifier value of the Aggregator that this Aggregation Port has currently selected. Zero indicates that the Aggregation Port has not selected an Aggregator, either because it is in the process of detaching from an Aggregator or because there is no suitable Aggregator available for it to select. This value is read-only.;

6.3.2.1.13 aAggPortAttachedAggID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The identifier value of the Aggregator to which this Aggregation Port is currently attached. Zero indicates that the Aggregation Port is not currently attached to an Aggregator. This value is read-only.;

6.3.2.1.14 aAggPortActorPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The port number locally assigned to the Aggregation Port. The port number is communicated in LACPDUs as the Actor_Port. This value is read-only.;

6.3.2.1.15 aAggPortActorPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The priority value assigned to this Aggregation Port. This 16-bit value is read-write.;

6.3.2.1.16 aAggPortPartnerAdminPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current administrative value of the port number for the protocol Partner. This is a 16-bit read-write value. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, and aAggPortPartnerAdminPortPriority, in order to achieve manually configured aggregation.;

6.3.2.1.17 aAggPortPartnerOperPort

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The operational port number assigned to this Aggregation Port by the Aggregation Port's protocol Partner. The value of this attribute may contain the manually configured value carried in aAggPortPartnerAdminPort if there is no protocol Partner. This 16-bit value is read-only.;

6.3.2.1.18 aAggPortPartnerAdminPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The current administrative value of the port priority for the protocol Partner. This is a 16-bit read-write value. The assigned value is used, along with the value of aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey, and aAggPortPartnerAdminPort, in order to achieve manually configured aggregation.;

6.3.2.1.19 aAggPortPartnerOperPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The priority value assigned to this Aggregation Port by the Partner. The value of this attribute may contain the manually configured value carried in aAggPortPartnerAdminPortPriority if there is no protocol Partner. This 16-bit value is read-only.;

6.3.2.1.20 aAggPortActorAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOUR DEFINED AS:

A string of 8 bits, corresponding to the administrative values of Actor_State (5.4.2) as transmitted by the Actor in LACPDUs. The first bit corresponds to bit 0 of Actor_State (LACP_Activity), the second bit corresponds to bit 1 (LACP_Timeout), the third bit corresponds to bit 2 (Aggregation), the fourth bit corresponds to bit 3 (Synchronization), the fifth bit corresponds to bit 4 (Collecting), the sixth bit corresponds to bit 5 (Distributing), the seventh bit corresponds to bit 6 (Defaulted), and the eighth bit corresponds to bit 7 (Expired). These values allow administrative control over the values of LACP_Activity, LACP_Timeout, and Aggregation. This attribute value is read-write.;

6.3.2.1.21 aAggPortActorOperState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOUR DEFINED AS:

A string of 8 bits, corresponding to the current operational values of Actor_State (5.4.2) as transmitted by the Actor in LACPDUs. The bit allocations are as defined in 6.3.2.1.20. This attribute value is read-only.;

6.3.2.1.22 aAggPortPartnerAdminState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOUR DEFINED AS:

A string of 8 bits, corresponding to the current administrative value of Actor_State for the protocol Partner. The bit allocations are as defined in 6.3.2.1.20. This attribute value is read-write. The assigned value is used in order to achieve manually configured aggregation.;

6.3.2.1.23 aAggPortPartnerOperState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOUR DEFINED AS:

A string of 8 bits, corresponding to the current values of Actor_State in the most recently received LACPDU transmitted by the protocol Partner. The bit allocations are as defined in 6.3.2.1.20. In the absence of an active protocol Partner, this value may reflect the manually configured value aAggPortPartnerAdminState. This attribute value is read-only.;

6.3.2.1.24 aAggPortAggregateOrIndividual

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOUR DEFINED AS:

A read-only Boolean value indicating whether the Aggregation Port is able to Aggregate (“TRUE”) or is only able to operate as an Individual link (“FALSE”).;

6.3.3 Aggregation Port Statistics managed object class

This subclause formally defines the behaviours for the oAggPortStats managed object class attributes.

6.3.3.1 Aggregation Port Statistics attributes

6.3.3.1.1 aAggPortStatsID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

This read-only attribute identifies an Aggregation Port Statistics object instance among the subordinate managed objects of the containing object. The value allocated to this attribute shall be the same as the containing oAggregationPort managed object.;

6.3.3.1.2 aAggPortStatsLACPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of valid LACPDUs received on this Aggregation Port. This value is read-only.;

6.3.3.1.3 aAggPortStatsMarkerPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of valid Marker PDUs received on this Aggregation Port. This value is read-only.;

6.3.3.1.4 aAggPortStatsMarkerResponsePDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of valid Marker Response PDUs received on this Aggregation Port. This value is read-only.;

6.3.3.1.5 aAggPortStatsUnknownRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 50 counts per second.

BEHAVIOUR DEFINED AS:

The number of frames received that either

—Carry the Slow Protocols Ethernet Type value (IEEE Std 802.3 Annex 57A.4), but contain an unknown PDU, or

—Are addressed to the Slow Protocols group MAC Address (IEEE Std 802.3 Annex 57A.3), but do not carry the Slow Protocols Ethernet Type. This value is read-only.;

6.3.3.1.6 aAggPortStatsIllegalRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 50 counts per second.

BEHAVIOUR DEFINED AS:

The number of frames received that carry the Slow Protocols Ethernet Type value (IEEE Std 802.3 Annex 57A.4), but contain a badly formed PDU or an illegal value of Protocol Subtype (IEEE Std 802.3 Annex 57A.3). This value is read-only.;

6.3.3.1.7 aAggPortStatsLACPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of LACPDUs transmitted on this Aggregation Port. This value is read-only.;

6.3.3.1.8 aAggPortStatsMarkerPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of Marker PDUs transmitted on this Aggregation Port. This value is read-only.;

6.3.3.1.9 aAggPortStatsMarkerResponsePDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of Marker Response PDUs transmitted on this Aggregation Port. This value is read-only.;

6.3.4 Aggregation Port Debug Information managed object class

This subclause formally defines the behaviours for the oAggPortDebugInformation managed object class attributes.

6.3.4.1 Aggregation Port Debug Information attributes

6.3.4.1.1 aAggPortDebugInformationID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

This read-only attribute identifies an LACP Debug Information object instance among the subordinate managed objects of the containing object. The value allocated to this attribute shall be the same as the containing oAggregationPort managed object.;

6.3.4.1.2 aAggPortDebugRxState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
current
expired
defaulted
initialize
lacpDisabled
portDisabled

BEHAVIOUR DEFINED AS:

This attribute holds the value “current” if the Receive state machine for the Aggregation Port is in the CURRENT state, “expired” if the Receive state machine is in the EXPIRED state, “defaulted” if the Receive state machine is in the DEFAULTED state, “initialize” if the Receive state machine is in the INITIALIZE state, “lacpDisabled” if the Receive state machine is in the LACP_DISABLED state, or “portDisabled” if the Receive state machine is in the PORT_DISABLED state. This value is read-only.;

6.3.4.1.3 aAggPortDebugLastRxTime

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The value of aTimeSinceSystemReset (See IEEE Std 802.3, F.2.1) when the last LACPDU was received by this Aggregation Port. This value is read-only.;

6.3.4.1.4 aAggPortDebugMuxState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
detached
waiting
attached
collecting
distributing
collecting_distributing

BEHAVIOUR DEFINED AS:

This attribute holds the value “detached” if the Mux state machine (5.4.15) for the Aggregation Port is in the DETACHED state, “waiting” if the Mux state machine for the Aggregation Port is in the WAITING state, “attached” if the Mux state machine for the Aggregation Port is in the ATTACHED state, “collecting” if the Mux state machine for the Aggregation Port is in the COLLECTING state, “distributing” if the Mux state machine for the Aggregation Port is in the DISTRIBUTING state, and “collecting_distributing” if the Mux state machine for the Aggregation Port is in the COLLECTING_DISTRIBUTING state. This value is read-only.;

6.3.4.1.5 aAggPortDebugMuxReason

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOUR DEFINED AS:

A human-readable text string indicating the reason for the most recent change of Mux machine state. This value is read-only.;

6.3.4.1.6 aAggPortDebugActorChurnState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
noChurn
churn

BEHAVIOUR DEFINED AS:

The state of the Actor Churn Detection machine (5.4.17) for the Aggregation Port. A value of "noChurn" indicates that the state machine is in either the NO_ACTOR_CHURN or the ACTOR_CHURN_MONITOR state, and "churn" indicates that the state machine is in the ACTOR_CHURN state. This value is read-only.;

6.3.4.1.7 aAggPortDebugPartnerChurnState

ATTRIBUTE

APPROPRIATE SYNTAX:

An ENUMERATED VALUE that has one of the following entries:
noChurn
churn

BEHAVIOUR DEFINED AS:

The state of the Partner Churn Detection machine (5.4.17) for the Aggregation Port. A value of "noChurn" indicates that the state machine is in either the NO_PARTNER_CHURN or the PARTNER_CHURN_MONITOR state, and "churn" indicates that the state machine is in the PARTNER_CHURN state. This value is read-only.;

6.3.4.1.8 aAggPortDebugActorChurnCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

Count of the number of times the Actor Churn state machine has entered the ACTOR_CHURN state. This value is read-only.;

6.3.4.1.9 aAggPortDebugPartnerChurnCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

Count of the number of times the Partner Churn state machine has entered the PARTNER_CHURN state. This value is read-only.;

6.3.4.1.10 aAggPortDebugActorSyncTransitionCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

Count of the number of times the Actor's Mux state machine (5.4.15) has entered the IN_SYNC state. This value is read-only.;

6.3.4.1.11 aAggPortDebugPartnerSyncTransitionCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

Count of the number of times the Partner's Mux state machine (5.4.15) has entered the IN_SYNC state. This value is read-only.;

6.3.4.1.12 aAggPortDebugActorChangeCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

Count of the number of times the Actor's perception of the LAG ID for this Aggregation Port has changed. This value is read-only.;

6.3.4.1.13 aAggPortDebugPartnerChangeCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized nonresetable counter. This counter has a maximum increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

Count of the number of times the Partner's perception of the LAG ID (5.3.6.1) for this Aggregation Port has changed. This value is read-only.;

Annex A

(informative)

Collection and Distribution functions

A.1 Introduction

The specification of the Collection and Distribution functions was defined with the following considerations in mind:

a) Frame duplication is not permitted.

b) Frame ordering must be preserved in aggregated links. Strictly, the MAC service specification (ISO/IEC 15802-1) states that order must be preserved for frames with a given SA, DA, and priority; however, this is a tighter constraint than is absolutely necessary. There may be multiple, logically independent conversations in progress between a given SA-DA pair at a given priority; the real requirement is to maintain ordering within a conversation, though not necessarily between conversations.

c) A single algorithm can be defined for the collection function that is independent of the distribution function(s) employed by the Partner System.

d) In the interests of simplicity and scalability, the collection function should not perform reassembly functions, reorder received frames, or modify received frames. Distribution functions, therefore, do not make use of segmentation techniques, do not label or otherwise modify transmitted frames in any way, and must operate in a manner that will inherently ensure proper ordering of received frames with the specified Collector.

e) The distribution and collection functions need to be capable of handling dynamic changes in aggregation membership.

f) There are expected to be many different topologies and many different types of devices in which Link Aggregation will be employed. It is therefore unlikely that a single distribution function will be applicable in all cases.

A simple collection function has been specified. The Collector preserves the order of frames received on a given link, but does not preserve frame ordering among links. The distribution function maintains frame ordering by

— Transmitting frames of a given conversation on a single link at any time.

— Before changing the link on which frames of a given conversation are transmitted, ensuring that all previously transmitted frames of that conversation have been received to a point such that any subsequently transmitted frames received on a different links will be delivered to the MAC Client at a later time.

Given the wide variety of potential distribution algorithms, the normative text in Clause 5 specifies only the requirements that such algorithms must meet, and not the details of the algorithms themselves. To clarify the intent, this informative annex gives examples of distribution algorithms, when they might be used, and the role of the Marker protocol (5.5) in their operation. The examples are not intended to be either exhaustive or prescriptive; implementers may make use of any distribution algorithms as long as the requirements of Clause 5 are met.

A.2 Port selection

A distribution algorithm selects the port used to transmit a given frame, such that the same port will be chosen for subsequent frames that form part of the same conversation. The algorithm may make use of information carried in the frame in order to make its decision, in combination with other information associated with the frame, such as its reception port in the case of a bridge.

The algorithm may assign one or more conversations to the same port; however, it must not allocate some of the frames of a given conversation to one port and the remainder to different ports. The information used to assign conversations to ports could include the following:

a) Source MAC address
b) Destination MAC address
c) The reception port
d) The type of destination address (individual or group MAC address)
e) Ethernet Length/Type value (i.e., protocol identification)
f) Higher layer protocol information (e.g., addressing and protocol identification information from the LLC sublayer or above)
g) Combinations of the above

One simple approach applies a hash function to the selected information to generate a port number. This produces a deterministic (i.e., history independent) port selection across a given number of ports in an aggregation. However, as it is difficult to select a hash function that will generate a uniform distribution of load across the set of ports for all traffic models, it might be appropriate to weight the port selection in favor of ports that are carrying lower traffic levels. In more sophisticated approaches, load balancing is dynamic; i.e., the port selected for a given set of conversations changes over time, independent of any changes that take place in the membership of the aggregation.

A.3 Dynamic reallocation of conversations to different ports

It may be necessary for a given conversation or set of conversations to be moved from one port to one or more others, as a result of

a) An existing port being removed from the aggregation,
b) A new port being added to the aggregation, or
c) A decision on the part of the Distributor to re-distribute the traffic across the set of ports.

Before moving conversation(s) to a new port, it is necessary to ensure that all frames already transmitted that are part of those conversations have been successfully received. The following procedure shows how the Marker protocol (5.5) can be used to ensure that no misordering of frames occurs:

1) Stop transmitting frames for the set of conversations affected. If the MAC Client requests transmission of further frames that are part of this set of conversations, these frames are discarded.
2) Start a timer, choosing the timeout period such that, if the timer expires, the destination System can be assumed either to have received or discarded all frames transmitted prior to starting the timer.
3) Use the Marker protocol to send a Marker PDU on the port previously used for this set of conversations.
4) Wait until either the corresponding Marker Response PDU is received or the timer expires.
5) Restart frame transmission for the set of conversations on the newly selected port.

The appropriate timeout value depends on the connected devices. For example, the recommended maximum Bridge Transit Delay is 1 s; if the receiving device is a bridge, it may be expected to have forwarded or

discarded all frames received more than 1 s ago. The appropriate timeout value for other circumstances could be smaller or larger than this by several orders of magnitude. For example, if the two Systems concerned are high-performance end stations connected via Gigabit Ethernet links, then timeout periods measured in milliseconds might be more appropriate. In order to allow an appropriate timeout value to be determined, the Frame Collector parameter CollectorMaxDelay (see 5.2.3) defines the maximum delay that the Collector can introduce between receiving a frame from a port and either delivering it to the MAC Client or discarding it. This value will be dependent upon the particular implementation choices that have been made in a System. As far as the operation of the Collector state machine is concerned, CollectorMaxDelay is a constant; however, a management attribute, aAggCollectorMaxDelay (6.3.1.1.32), is provided that allows interrogation and administrative control of its value. Hence, if a System knows the value of CollectorMaxDelay that is in use by a Partner System, it can set the value of timeout used when flushing a link to be equal to that value of CollectorMaxDelay, plus sufficient additional time to allow for the propagation delay experienced by frames between the two Systems. A value of zero for the CollectorMaxDelay parameter indicates that the delay imposed by the Collector is less than the resolution of the parameter (10 μs). In this case, the delay that must be considered is the physical propagation delay of the channel. Allowing management manipulation of CollectorMaxDelay permits fine-tuning of the value used in those cases where it may be difficult for the equipment to pre-configure a piece of equipment with a realistic value for the physical propagation delay of the channel.

The Marker protocol provides an optimization that can result in faster reallocation of conversations than would otherwise be possible—without the use of markers, the full timeout period would always have to be used in order to be sure that no frames remained in transit between the local Distributor and the remote Collector. The timeout described recovers from loss of Marker or Marker Response PDUs that can occur.

A.4 Topology considerations in the choice of distribution algorithm

Figure A–1 gives some examples of different aggregated link scenarios. In some cases, it is possible to use distribution algorithms that use MAC frame information to allocate conversations to links; in others, it is necessary to make use of higher-layer information.

In example A, there is a many-to-many relationship between end stations communicating over the aggregated link. It would be possible for each switch to allocate conversations to links simply on the basis of source or destination MAC addresses.

In examples B and C, a number of end stations communicate with a single server via the aggregated link. In these cases, the distribution algorithm employed in the server or in Switch 2 can allocate traffic from the server on the basis of destination MAC address; however, as one end of all conversations constitutes a single server with a single MAC address, traffic from the end stations to the server would have to be allocated on the basis of source MAC address. These examples illustrate the fact that different distribution algorithms can be used in different devices, as appropriate to the circumstances. The collection function is independent of the distribution function(s) that are employed.

In examples D and E, assuming that the servers are using a single MAC address for all of their traffic, the only appropriate option is for the distribution algorithm used in the servers and switches to make use of higher-layer information (e.g., Transport Layer socket identifiers) in order to allocate conversations to links. Alternatively, in example E, if the servers were able to make use of multiple MAC addresses and allocate conversations to them, then the switches could revert to MAC Address-based allocation.

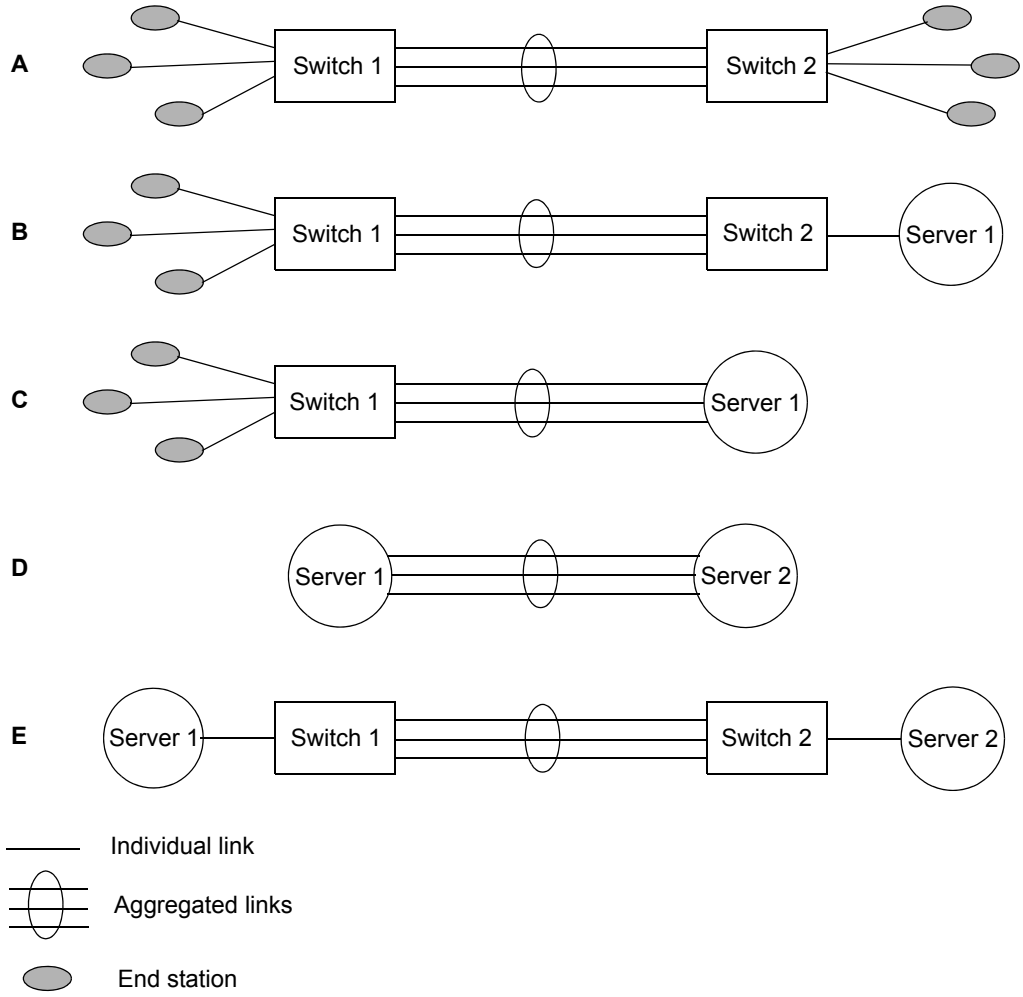


Figure A–1—Link aggregation topology examples

Annex B

(informative)

LACP standby link selection and dynamic Key management

B.1 Introduction

While any two ports on a given system that have been assigned the same administrative Key may be capable of aggregation, it is not necessarily the case that an arbitrary selection of such ports can be aggregated. (Keys may have been deliberately assigned to allow one link to be operated specifically as a hot standby for another.) A system may reasonably limit the number of ports attached to a single Aggregator, or the particular way more than two ports can be combined.

In cases where both communicating systems have constraints on aggregation, it is necessary for them both to agree to some extent on the links to be selected for aggregation and on which not to use. Otherwise it might be possible for the two systems to make different selections, possibly resulting in no communication at all.

When one or more links have to be selected as standby, it is possible that they could be used as part of a different Link Aggregation Group. For this to happen, one or another of the communicating systems has to change the operational Key values used for the ports attached to those links.

If the operational Key values were to be changed independently by each system, the resulting set of aggregations could be unpredictable. It is possible that numerous aggregations, each containing a single link, may result. Worse, with no constraint on changes, the process of both systems independently searching for the best combination of operational Key values may never end.

This annex describes protocol rules for standby link selection and dynamic Key management. It provides examples of a dynamic Key management algorithm applied to connections between systems with various aggregation constraints.

B.2 Goals

The protocol rules presented

a) Enable coordinated, predictable, and reproducible standby link selections.
b) Permit predictable and reproducible partitioning of links into aggregations by dynamic Key management.

They do not require

c) A LACP system to understand all the constraints on aggregations of multiple ports that might be imposed by other systems.
d) Correct configuration of parameters, i.e., they retain the plug and play attributes of LACP.

B.3 Standby link selection

Every link between systems operating LACP is assigned a unique priority. This priority comprises (in priority order) the System Priority, System ID, Port Priority, and Port Number of the higher-priority system. In priority comparisons, numerically lower values have higher priority.

Ports are considered for active use in an aggregation in link priority order, starting with the port attached to the highest priority link. Each port is selected for active use if preceding higher priority selections can also be maintained; otherwise, the port is selected as standby.

B.4 Dynamic Key management

Dynamic Key management changes the Key values used for links that either system has selected as a standby to allow use of more links. Whether this is desirable depends on their use. For example, if a single spanning tree is being used throughout the network, separating standby links into a separate aggregation serves little purpose. In contrast, if equal cost load sharing is being provided by routing, making additional bandwidth available in a separate Link Aggregation Group may be preferable to holding links in standby to provide link resilience.

The communicating system with the higher priority (as determined by System Priority and unique System ID) controls dynamic Key changes. Dynamic Key changes may only be made by this controlling system.

NOTE—The controlling system can observe the port priorities assigned by the Partner system, if it wishes to take these into account.

This rule prevents the confusion that could arise if both systems change Keys simultaneously. In principle the controlling system might search all possible Key combinations for the best way to partition the links into groups. In practice the number of times that Keys may have to be changed to yield acceptable results is small.

After each Key change, the controlling system assesses which links are being held in standby by its Partner. Although there is no direct indication of this decision, standby links will be held OUT_OF_SYNC. After matched information is received from the protocol Partner, and before acting on this information, a “settling time” allows for the Partner’s aggregate wait delay, and for the selected links to be aggregated. Twice the Aggregate Wait Time (the expiry period for the wait_while_timer), i.e., 4 s, should be ample. If matched Partner information indicates that all the links that the Actor can make active have been brought IN_SYNC, it can proceed to change Keys on other links without further delay.

B.5 A dynamic Key management algorithm

The following algorithm is simple but effective.

After the “settling time” (see B.4) has elapsed, the controlling system scans its ports in the Link Aggregation Group (i.e., all those ports with a specific operational Key value that have the same Partner System Priority, System ID, and Key) in descending priority order.

For each port, it may wish to know

a) Is the port (i.e., the Actor) *capable* of being aggregated with the ports already selected for aggregation with the current Key? Alternatively, is the Actor *not capable* of this aggregation?

b) Is the port’s Partner IN_SYNC or is the Partner OUT_OF_SYNC?

As it inspects each port, it may

c) *Select* the port to be part of the aggregation with the current Key.

d) *Retain* the current Key for a further iteration of the algorithm, without selecting the port to be part of the current aggregation.

e) *Change* the operational Key to a new value. Once a new value is chosen, all the ports in the current Link Aggregation Group that have their Keys changed will be changed to this new value.

As the ports are scanned for the first time

1) The highest priority port is always selected.

   If it is capable and IN_SYNC, move to step 2).

   Otherwise, **change** the operational Key of all other ports (if any) in this Link Aggregation Group, and apply this dynamic Key algorithm to those ports, beginning with step 1), after the settling time.

2) Move to the next port.

   IF there is a next port, continue at step 3).

   Otherwise, dynamic Key changes for ports with this operational Key are complete.

   Note that ports that were once in the same aggregation may have had their operational Keys changed to (further) new values. If so, apply the dynamic Key management algorithms to those ports, beginning with step 1), after the settling time.

3) If this port is capable and IN_SYNC:

      select it, and repeat from step 2).

   If this port is OUT_OF_SYNC:

      change the operational Key, and repeat from step 2).

   If this port is not capable but IN_SYNC:

      change the operational Key, move to step 4).

4) Move to the next port.

   If there is a next port, continue at step 5).

   Otherwise If there are still ports in the current Link Aggregation Group (which will have the current operational Key), wait for the settling time and apply the dynamic Key management algorithm, beginning with the first such port, at step 3).

   Otherwise, dynamic Key changes for ports with this operational Key are complete.

5) **If** this port is capable:

      retain the current Key and repeat from step 2).

   Otherwise, **change** the operational Key and repeat from step 2).

This procedure is repeated until no OUT_OF_SYNC links remain, or a limit on the number of steps has been reached.

If the Partner's System ID changes on any link at any time, the Actor's operational Key for that link should revert to the administrative Key value, and the dynamic Key procedure should be rerun. This may involve changing the operational Key values for all the links that were assigned Key values subsequent to the change in Key for the link with the new Partner.

B.6 Example 1

Two systems, A and B, are connected by four parallel links. Each system can support a maximum of two links in an aggregation. They are connected as shown in Figure B–1. System A is the higher priority system.

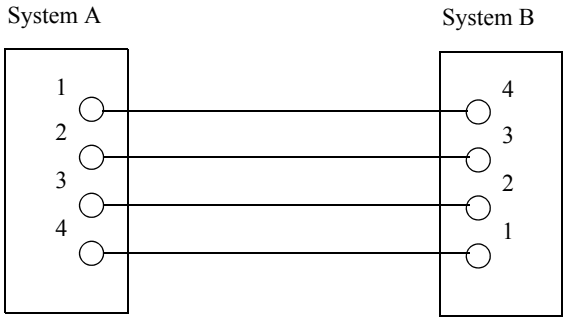


Figure B–1—Example 1

The administrative Key for all of System A and System B's ports is 1. Neither system knows before the configuration is chosen that all its ports would attach to links of the same Partner system. Equally, if the links were attached to two different systems, it is not known which pair of links (e.g., 1 and 2, or 1 and 4) would be attached to the same Partner. So choosing the administrative Keys values to be identical for four ports, even though only two could be actively aggregated, is very reasonable.

If there was no rule for selecting standby links, System A and System B might have both selected their own ports 1 and 2 as the active links, and there would be no communication. With the rule, the links A1-B4 and A2-B3 will become active, while A3-B2 and A4-B1 will be standby.

Since System A is the higher-priority system, System B's operational Key values will remain 1 while System A may dynamically change Keys, though it may choose to retain the standby links. Following the Key management algorithm suggested, System A would be able to change the Keys for A3 and A4 in a little over 2 seconds (depending on how fast System B completes the process of attaching its ports to the selected Aggregator) after the connections were first made, and both aggregations could be operating within 5 seconds.

If System A's aggregations were to be constrained to a maximum of three links, rather than two, while System B's are still constrained to two, the suggested algorithm would delay for 4 s before changing Keys. Both aggregations could be operating within 7 s.

B.7 Example 2

A system has the odd design constraint that each of its four ports may be aggregated with one other as follows:

a) Port 1 with port 2, or port 4.
b) Port 2 with port 3, or port 1.
c) Port 3 with port 4, or port 2.
d) Port 4 with port 1, or port 3.

This is equivalent to each port being able to aggregate with either neighbor, understanding the ports to be arranged in a circle.

Two such systems are connected with four parallel links as shown in Figure B–2.

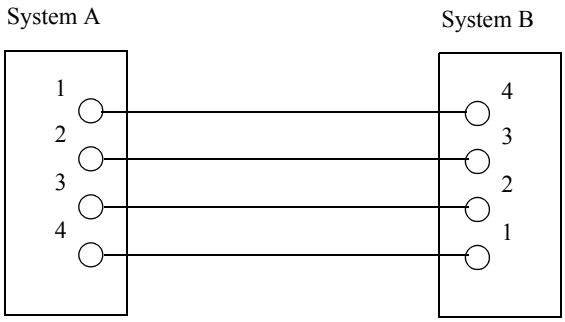


Figure B–2—Example 2a

Just as for Example 1, links A1-B4 and A2-B3 become active without changing the operational Key from its original administrative value. The Key for A3 and A4 is changed as soon as they become active, and a few seconds later A3-B2 and A4-B1 become active in a separate aggregation.

If the two systems had been connected as shown in Figure B–3:

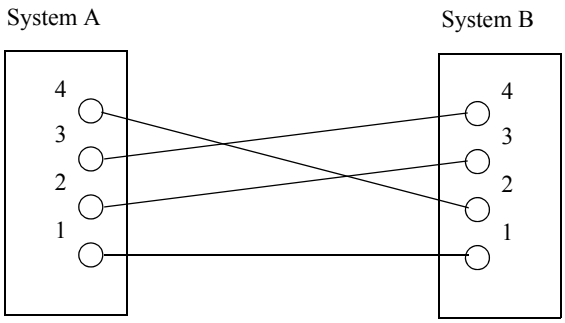


Figure B–3—Example 2b

the following would occur, assuming that System A operates the simple algorithm already described.

Initially System A advertises an operational Key equal to the administrative Key value of 1 on all ports. System B first selects B1 as active; since the link connects to A1 it has the highest priority. The next highest priority link is B3-A2, but System B cannot aggregate B3 with B1, so System B makes this port standby. System B can aggregate B4-A3, so the port is made active. Finally if B4 is aggregated with B1, B2 cannot be aggregated, so B2 is made standby.

System A, observing the resulting synchronization status from System B, assigns a Key value of 2 to ports 2 and 3, retaining the initial Key of 1 for ports 1 and 4. System B will remove B4 from the aggregation with B1, and substitute B2. B3 and B4 will be aggregated. In the final configuration A1-B1 and A4-B2 are aggregated, as are A2-B3 and A3-B4.

Annex C

(normative)

SNMP MIB definitions for Link Aggregation

C.1 Introduction

This annex defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP based internets. In particular it defines objects for managing the operation of the Link Aggregation sublayer, based on the specification of Link Aggregation contained in Clause 6. This annex includes an MIB module that is SNMPv2 SMI compliant.

C.2 The SNMP Management Framework

The SNMP Management Framework presently consists of five major components

a) An overall architecture, described in IETF RFC 2271.

b) Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIv1 and is described in IETF RFC 1155, IETF RFC 1212, and IETF RFC 1215. The second version, called SMIv2, is described in IETF RFC 1902, IETF RFC 1903, and IETF RFC 1904.

c) Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and is described in IETF RFC 1157. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and is described in IETF RFC 1901 and IETF RFC 1906. The third version of the message protocol is called SNMPv3 and is described in IETF RFC 1906, IETF RFC 2272, and IETF RFC 2274.

d) Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in IETF RFC 1157. A second set of protocol operations and associated PDU formats is described in IETF RFC 1905.

e) A set of fundamental applications described in IETF RFC 2273 and the view-based access control mechanism described in IETF RFC 2275.

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This annex specifies an MIB module that is compliant to the SMIv2. An MIB conforming to the SMIv1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (use of Counter64). Some machine-readable information in SMIv2 will be converted into textual descriptions in SMIv1 during the translation process. However, this loss of machine-readable information is not considered to change the semantics of the MIB.

C.3 Security considerations

There are a number of management objects defined in this MIB 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 non-secure environment can have a negative effect on network operations.

SNMPv1 by itself is not a secure environment. Even if the network itself is secure (e.g., by using IPSec), there is no control as to who on the secure network is allowed to access (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model IETF RFC 2274 and the View-based Access Control Model IETF RFC 2275 is recommended. It then becomes a user responsibility to ensure that the SNMP entity giving access to an instance of this MIB is properly configured to give access only to those principals (users) that have legitimate rights to access (read/change/create/delete) them, as appropriate.

C.4 Structure of the MIB

A single MIB module is defined in this annex. Objects in the MIB are arranged into groups. Each group is organized as a set of related objects. The overall structure and assignment of objects to their groups is shown in the following subclauses.

C.4.1 Relationship to the managed objects defined in Clause 6

This subclause contains cross-references to the objects defined in Clause 6. Table C–1 contains cross references for the MIB objects defined in this annex. Table C–2 contains definitions of ifTable elements, as defined in IETF RFC 2233, for an Aggregator. These table elements are cross referenced to the corresponding definitions in Clause 6.

Table C–1—Managed object cross reference table

Definition in Clause 6	MIB object
6.3.1.1.1 aAggID	ifIndex value (see IETF RFC 2233)
6.3.1.1.4 aAggActorSystemID	dot3adAggActorSystemID
6.3.1.1.5 aAggActorSystemPriority	dot3adAggActorSystemPriority
6.3.1.1.6 aAggAggregateOrIndividual	dot3adAggAggregateOrIndividual
6.3.1.1.7 aAggActorAdminKey	dot3adAggActorAdminKey
6.3.1.1.8 aAggActorOperKey	dot3adAggActorOperKey
6.3.1.1.10 aAggPartnerSystemID	dot3adAggPartnerSystemID
6.3.1.1.11 aAggPartnerSystemPriority	dot3adAggPartnerSystemPriority
6.3.1.1.12 aAggPartnerOperKey	dot3adAggPartnerOperKey
6.3.1.1.30 aAggPortList	dot3adAggPortListTable
6.3.1.1.32 aAggCollectorMaxDelay	dot3adAggCollectorMaxDelay
6.3.2.1.1 aAggPortID	ifIndex value (see IETF RFC 2233)
6.3.2.1.2 aAggPortActorSystemPriority	dot3adAggPortActorSystemPriority
6.3.2.1.3 aAggPortActorSystemID	dot3adAggPortActorSystemID
6.3.2.1.4 aAggPortActorAdminKey	dot3adAggPortActorAdminKey
6.3.2.1.5 aAggPortActorOperKey	dot3adAggPortActorOperKey

Table C–1—Managed object cross reference table *(continued)*

Definition in Clause 6	MIB object
6.3.2.1.6 aAggPortPartnerAdminSystemPriority	dot3adAggPortPartnerAdminSystemPriority
6.3.2.1.7 aAggPortPartnerOperSystemPriority	dot3adAggPortPartnerOperSystemPriority
6.3.2.1.8 aAggPortPartnerAdminSystemID	dot3adAggPortPartnerAdminSystemID
6.3.2.1.9 aAggPortPartnerOperSystemID	dot3adAggPortPartnerOperSystemID
6.3.2.1.10 aAggPortPartnerAdminKey	dot3adAggPortPartnerAdminKey
6.3.2.1.11 aAggPortPartnerOperKey	dot3adAggPortPartnerOperKey
6.3.2.1.12 aAggPortSelectedAggID	dot3adAggPortSelectedAggID
6.3.2.1.13 aAggPortAttachedAggID	dot3adAggPortAttachedAggID
6.3.2.1.14 aAggPortActorPort	dot3adAggPortActorPort
6.3.2.1.15 aAggPortActorPortPriority	dot3adAggPortActorPortPriority
6.3.2.1.16 aAggPortPartnerAdminPort	dot3adAggPortPartnerAdminPort
6.3.2.1.17 aAggPortPartnerOperPort	dot3adAggPortPartnerOperPort
6.3.2.1.18 aAggPortPartnerAdminPortPriority	dot3adAggPortPartnerAdminPortPriority
6.3.2.1.19 aAggPortPartnerOperPortPriority	dot3adAggPortPartnerOperPortPriority
6.3.2.1.20 aAggPortActorAdminState	dot3adAggPortActorAdminState
6.3.2.1.21 aAggPortActorOperState	dot3adAggPortActorOperState
6.3.2.1.22 aAggPortPartnerAdminState	dot3adAggPortPartnerAdminState
6.3.2.1.23 aAggPortPartnerOperState	dot3adAggPortPartnerOperState
6.3.2.1.24 aAggPortAggregateOrIndividual	dot3adAggPortAggregateOrIndividual
6.3.3.1.1 aAggPortStatsID	ifIndex value (see IETF RFC 2233) of the port
6.3.3.1.2 aAggPortStatsLACPDUsRx	dot3adAggPortStatsLACPDUsRx
6.3.3.1.3 aAggPortStatsMarkerPDUsRx	dot3adAggPortStatsMarkerPDUsRx
6.3.3.1.4 aAggPortStatsMarkerResponsePDUsRx	dot3adAggPortStatsMarkerResponsePDUsRx
6.3.3.1.5 aAggPortStatsUnknownRx	dot3adAggPortStatsUnknownRx
6.3.3.1.6 aAggPortStatsIllegalRx	dot3adAggPortStatsIllegalRx
6.3.3.1.7 aAggPortStatsLACPDUsTx	dot3adAggPortStatsLACPDUsTx
6.3.3.1.8 aAggPortStatsMarkerPDUsTx	dot3adAggPortStatsMarkerPDUsTx
6.3.3.1.9 aAggPortStatsMarkerResponsePDUsTx	dot3adAggPortStatsMarkerResponsePDUsTx
6.3.4.1.1 aAggPortDebugInformationID	ifIndex value (see IETF RFC 2233) of the port
6.3.4.1.2 aAggPortDebugRxState	dot3adAggPortDebugRxState
6.3.4.1.3 aAggPortDebugLastRxTime	dot3adAggPortDebugLastRxTime

Table C–1—Managed object cross reference table *(continued)*

Definition in Clause 6	MIB object
6.3.4.1.4 aAggPortDebugMuxState	dot3adAggPortDebugMuxState
6.3.4.1.5 aAggPortDebugMuxReason	dot3adAggPortDebugMuxReason
6.3.4.1.6 aAggPortDebugActorChurnState	dot3adAggPortDebugActorChurnState
6.3.4.1.7 aAggPortDebugPartnerChurnState	dot3adAggPortDebugPartnerChurnState
6.3.4.1.8 aAggPortDebugActorChurnCount	dot3adAggPortDebugActorChurnCount
6.3.4.1.9 aAggPortDebugPartnerChurnCount	dot3adAggPortDebugPartnerChurnCount
6.3.4.1.10 aAggPortDebugActorSyncTransitionCount	dot3adAggPortDebugActorSyncTransitionCount
6.3.4.1.11 aAggPortDebugPartnerSyncTransitionCount	dot3adAggPortDebugPartnerSyncTransitionCount
6.3.4.1.12 aAggPortDebugActorChangeCount	dot3adAggPortDebugActorChangeCount
6.3.4.1.13 aAggPortDebugPartnerChangeCount	dot3adAggPortDebugPartnerChangeCount

Table C–2—ifTable element definitions for an Aggregator

Object	Definition
ifIndex	A unique integer value is allocated to each Aggregator by the local System. Interpreted as defined in IETF RFC 2233.
ifDescr	Interpreted as defined in IETF RFC 2233 and as further refined in the definition of aAggDescription (6.3.1.1.2).
ifType	ieee8023adLag(161)[a].
ifMTU	The largest MAC Client SDU that can be carried by this Aggregator—1500 octets.
ifSpeed	The data rate of the Aggregation as defined for aAggDataRate (6.3.1.1.16).
ifPhysAddress	The individual MAC Address of the Aggregator as defined for aAggMACAddress (6.3.1.1.9).
ifAdminStatus	The administrative state of the Aggregator as defined for aAggAdminState (6.3.1.1.13).
ifOperStatus	The operational state of the Aggregator as defined for aAggOperState (6.3.1.1.14).
ifLastChange	Interpreted as defined in IETF RFC 2233; see also the definition of aAggTimeOfLastOperChange (6.3.1.1.15).
ifInOctets	The total number of user data octets received by the aggregation, as defined for aAggOctetsRxOK (6.3.1.1.18).
ifInUcastPkts	The total number of unicast user data frames received by the aggregation. This value is calculated as the value of aAggFramesRxOK (6.3.1.1.20), less the values of aAggMulticastFramesRxOK (6.3.1.1.22) and aAggBroadcastFramesRxOK (6.3.1.1.24).
ifInNUcastPkts	Deprecated in IETF RFC 2233.
ifInDiscards	The number of frames discarded on reception, as defined for aAggFramesDiscardedOnRx (6.3.1.1.26).

Table C–2—ifTable element definitions for an Aggregator *(continued)*

Object	Definition
ifInErrors	The number of frames with reception errors, as defined for aAggFramesWith-RxErrors (6.3.1.1.28).
ifInUnknownProtos	The number of unknown protocol frames discarded on reception, as defined for aAggUnknownProtocolFrames (6.3.1.1.29).
ifOutOctets	The total number of user data octets transmitted by the aggregation, as defined for aAggOctetsTxOK (6.3.1.1.17).
ifOutUcastPkts	The total number of unicast user data frames transmitted by the aggregation. This value is calculated as the value of aAggFramesTxOK (6.3.1.1.19), less the values of aAggMulticastFramesTxOK (6.3.1.1.21) and aAggBroadcast-FramesTxOK (6.3.1.1.23).
ifOutNUcastPkts	Deprecated in IETF RFC 2233.
ifOutDiscards	The number of frames discarded on transmission, as defined for aAggFrames-DiscardedOnTx (6.3.1.1.25).
ifOutErrors	The number of frames discarded due to transmission errors, as defined for aAggFramesWithTxErrors (6.3.1.1.27).
ifOutQLen	Deprecated in IETF RFC 2233. Set to zero if present.
ifSpecific	Deprecated in IETF RFC 2233. Set to { 0.0 } if present.
ifLinkUpDownTrapEnable	See the definition of aAggLinkUpDownNotificationEnable (6.3.1.1.31).
ifConnectorPresent	"FALSE."
ifHighSpeed	Set to zero.
ifName	The locally assigned textual name of the Aggregator, as defined for aAggName (6.3.1.1.3). Interpreted as defined in IETF RFC 2233.
linkUp TRAP	See the definition of nAggLinkUpNotification (6.3.1.2.1).
linkDown TRAP	See the definition of nAggLinkDownNotification (6.3.1.2.2).

[a]Values of ifType are assigned by the Internet Assigned Numbers Authority (IANA). A directory of number assignments is maintained on their website, at URL: http://www.iana.org/numbers.html. The currently assigned ifType values can be found in the SMI Numbers (Network Management Parameters) section of that directory.

C.4.2 The Aggregator Group

This group of objects, in combination with the ifTable entry for an Aggregator and the Aggregator Port List, provides the functionality of the Aggregator managed object class (6.3.1). The Aggregator Group provides the control elements necessary to configure an Aggregator, plus the statistical information necessary to monitor the behaviour of an Aggregator.

C.4.3 The Aggregator Port List Group

This group of objects implements the functionality defined for the Aggregator Port List attribute (6.3.1.1.30).

C.4.4 The Aggregation Port Group

This group of objects provides the functionality of the Aggregation Port managed object class (6.3.2). The Aggregation Port Group provides the control elements necessary to configure a Port for Link Aggregation, plus the statistical information necessary to monitor the behavior of the port.

C.4.5 The Aggregation Port Statistics Group

This group of objects provides the functionality of the Aggregation Port Statistics managed object class (6.3.3). The Aggregation Port Statistics Group provides additional statistical information related to LACP and Marker protocol activity on the port.

C.4.6 The Aggregation Port Debug Group

This group of objects provides the functionality of the Aggregation Port Debug managed object class (6.3.4). The Aggregation Port Debug Group provides additional information related to the operation of LACP on the port; this information is primarily aimed at debugging the operation of the protocol and detecting fault conditions.

C.5 Relationship to other MIBs

It is assumed that a system implementing this MIB will also implement (at least) the "system" group defined in MIB-II defined in IETF RFC 1213 and the "interfaces" group defined in IETF RFC 2233.

C.5.1 Relationship to the Interfaces MIB

IETF RFC 2233, the Interface MIB Evolution, requires that any MIB that is an adjunct of the Interface MIB, clarify specific areas within the Interface MIB. These areas were intentionally left vague in IETF RFC 2233 to avoid over constraining the MIB, thereby precluding management of certain media types.

Section 3.3 of IETF RFC 2233 enumerates several areas that a media-specific MIB must clarify. Each of these areas is addressed in C.5.2 and C.5.3. The implementer is referred to IETF RFC 2233 in order to understand the general intent of these areas.

In IETF RFC 2233, the "interfaces" group is defined as being mandatory for all systems and contains information on an entity's interfaces, where each 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 sometimes used to refer to such a subnetwork.

Implicit in this MIB is the notion of Aggregators and Aggregation ports. Each of these Aggregators and Aggregation ports is associated with one interface of the "interfaces" group (one row in the ifTable) and each port is associated with a different interface.

Each Aggregator and Aggregation port is uniquely identified by an interface number (ifIndex). The ifIndex value assigned to a given Aggregation port is the same as the ifIndex value assigned to the MAC interface with which that Aggregation port is associated.

C.5.2 Layering model

This annex assumes the interpretation of the Interfaces Group to be in accordance with IETF RFC 2233, which states that the 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 annex recommends that, within an entity, aggregations that are instantiated as an entry in dot3adAggTable are also represented by an entry in ifTable.

Where an entity contains Link Aggregation entities that transmit and receive traffic to/from an aggregation, these should be represented in the ifTable as interfaces of type ieee8023adLag(161).

C.5.3 ifStackTable

If the ifStackTable defined in IETF RFC1573 is implemented, then

a) The relationship shown in the table has the property that an Aggregation is a higher interface relative to an Aggregation Port.

b) This relationship is read-only.

NOTE—The restriction stated here is intended to enforce a strict hierarchical relationship between Aggregations and Aggregation Ports, and to prevent those relationships from being modified. The read-only restriction does not apply to any other relationships that may be expressed in the ifStackTable.

C.5.4 ifRcvAddressTable

The ifRcvAddressTable contains all MAC Addresses, unicast, multicast, and broadcast, for which an interface can receive frames and forward them up to a higher layer entity for local consumption. An Aggregator has at least one such address.

C.6 Definitions for Link Aggregation MIB

In the following MIB definition,[8] should there be any discrepancy between the DESCRIPTION text and the BEHAVIOUR DEFINED AS in the corresponding definition in Clause 6, the definition in Clause 6 shall take precedence.

NOTE—The ASCII for C.6 is available at http://www.ieee802.org/3/publication/index.html.[9]

[8]MIB definitions are available at http://www.ietf.org/.

[9]*Copyright release for SNMP MIB:* Users of this standard may freely reproduce the SNMP MIB in this annex so it can be used for its intended purpose.

```
IEEE8023-LAG-MIB DEFINITIONS ::= BEGIN


-- ------------------------------------------------------------
-- IEEE 802.3ad MIB
-- ------------------------------------------------------------


IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, TimeTicks, BITS
        FROM SNMPv2-SMI
    DisplayString, MacAddress, TEXTUAL-CONVENTION, TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    InterfaceIndex
        FROM IF-MIB
    PortList
        FROM Q-BRIDGE-MIB
    ;


lagMIB MODULE-IDENTITY
    LAST-UPDATED "200706290000Z"
    REVISION "Original publication IEEE 802.3ad"
    REVISION "References updated 04 Jun 2007 for IEEE 802.1AX"
    ORGANIZATION "IEEE 802.3ad Working Group"
    CONTACT-INFO
        " stds-8021@ieee.org"
    DESCRIPTION
        "The Link Aggregation module for managing IEEE 802.1AX."
    ::= { iso(1) member-body(2) us(840) ieee802dot3(10006) snmpmibs(300) link-
     agg(43) }


lagMIBObjects OBJECT IDENTIFIER ::= { lagMIB 1 }


-- ------------------------------------------------------------
-- Textual Conventions
-- ------------------------------------------------------------


LacpKey ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The Actor or Partner Key value."
    SYNTAX      INTEGER (0..65535)


LacpState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The Actor and Partner State values from the LACPDU."
    SYNTAX      BITS {
        lacpActivity(0),
        lacpTimeout(1),
        aggregation(2),
        synchronization(3),
```



```
        collecting(4),
        distributing(5),
        defaulted(6),
        expired(7)
                }


ChurnState ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The state of the Churn Detection machine."
    SYNTAX      INTEGER {
                    noChurn(1),
                    churn(2),
                    churnMonitor(3)
                }


-- ---------------------------------------------------------


-- ---------------------------------------------------------
-- groups in the LAG MIB
-- ---------------------------------------------------------


dot3adAgg OBJECT IDENTIFIER ::= { lagMIBObjects 1 }
dot3adAggPort OBJECT IDENTIFIER ::= { lagMIBObjects 2 }


-- ---------------------------------------------------------

-- ---------------------------------------------------------
-- The Tables Last Changed Object
-- ---------------------------------------------------------

dot3adTablesLastChanged OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the time of the
        most recent change to the dot3adAggTable,
        dot3adAggPortListTable, or
        dot3adAggPortTable."
::= { lagMIBObjects 3 }

-- ---------------------------------------------------------
-- The Aggregator Configuration Table
-- ---------------------------------------------------------


dot3adAggTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains information about every
        Aggregator that is associated with this System."
```

```
    REFERENCE
        "6.3.1"
    ::= { dot3adAgg 1 }


dot3adAggEntry OBJECT-TYPE
    SYNTAX      Dot3adAggEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of the Aggregator parameters. This is indexed
        by the ifIndex of the Aggregator."
    INDEX { dot3adAggIndex }
    ::= { dot3adAggTable 1 }


Dot3adAggEntry ::=
    SEQUENCE {
        dot3adAggIndex
            InterfaceIndex,
        dot3adAggMACAddress
            MacAddress,
        dot3adAggActorSystemPriority
            INTEGER,
        dot3adAggActorSystemID
            MacAddress,
        dot3adAggAggregateOrIndividual
            TruthValue,
        dot3adAggActorAdminKey
            LacpKey,
        dot3adAggActorOperKey
            LacpKey,
        dot3adAggPartnerSystemID
            MacAddress,
        dot3adAggPartnerSystemPriority
            INTEGER,
        dot3adAggPartnerOperKey
            LacpKey,
        dot3adAggCollectorMaxDelay
            INTEGER
    }


dot3adAggIndex OBJECT-TYPE
    SYNTAX       InterfaceIndex
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
       "The unique identifier allocated to this Aggregator by the local System.
        This attribute identifies an Aggregator instance among the subordinate
        managed objects of the containing object. This value is read-only."
    REFERENCE
        "6.3.1.1.1"
    ::= { dot3adAggEntry 1 }


dot3adAggMACAddress OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
```



```
    STATUS       current
    DESCRIPTION
        "A 6-octet read-only value carrying the individual
       MAC address assigned to the Aggregator."
    REFERENCE
        "6.3.1.1.9"
    ::= { dot3adAggEntry 2 }


dot3adAggActorSystemPriority OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A 2-octet read-write value indicating the priority value
        associated with the Actor's System ID."
    REFERENCE
        "6.3.1.1.5"
    ::= { dot3adAggEntry 3 }


dot3adAggActorSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 6-octet read-write MAC address value used as a unique
        identifier for the System that contains this Aggregator.
        NOTE-From the perspective of the Link Aggregation
        mechanisms described in Clause 5, only a single
        combination of Actor's System ID and System Priority are
        considered, and no distinction is made between the
        values of these parameters for an Aggregator and the
        port(s) that are associated with it; i.e., the protocol
        is described in terms of the operation of aggregation
        within a single System. However, the managed objects
        provided for the Aggregator and the port both allow
        management of these parameters. The result of this is to
        permit a single piece of equipment to be configured by
        management to contain more than one System from the
        point of view of the operation of Link Aggregation. This
        may be of particular use in the configuration of
        equipment that has limited aggregation capability (see 5.6)."
REFERENCE
        "6.3.1.1.4"
    ::= { dot3adAggEntry 4 }


dot3adAggAggregateOrIndividual OBJECT-TYPE
    SYNTAX       TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A read-only Boolean value indicating whether the
        Aggregator represents an Aggregate ('TRUE') or
        an Individual link ('FALSE')."
    REFERENCE
        "6.3.1.1.6"
    ::= { dot3adAggEntry 5 }
```

```
dot3adAggActorAdminKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The current administrative value of the Key for the Aggregator.
        The administrative Key value may differ from the operational
        Key value for the reasons discussed in 5.6.2. This is a 16-bit,
        read-write value. The meaning of particular Key values
        is of local significance."
    REFERENCE
        "6.3.1.1.7"
    ::= { dot3adAggEntry 6 }


dot3adAggActorOperKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current operational value of the Key for the Aggregator.
        The administrative Key value may differ from the operational
        Key value for the reasons discussed in 5.6.2.
        This is a 16-bit read-only value. The meaning of particular Key
        values is of local significance.
    REFERENCE
        "6.3.1.1.8"
    ::= { dot3adAggEntry 7 }


dot3adAggPartnerSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 6-octet read-only MAC address value consisting
        of the unique identifier for the current protocol Partner of
        this Aggregator. A value of zero indicates that there is no
        known Partner. If the aggregation is manually configured, this
        System ID value will be a value assigned by the local System."
    REFERENCE
        "6.3.1.1.10"
    ::= { dot3adAggEntry 8 }


dot3adAggPartnerSystemPriority OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 2-octet read-only value that indicates the priority
        value associated with the Partner's System ID. If the
        aggregation is manually configured, this System Priority value
        will be a value assigned by the local System."
    REFERENCE
        "6.3.1.1.11"
    ::= { dot3adAggEntry 9 }
```



```
dot3adAggPartnerOperKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current operational value of the Key for the
        Aggregator's current protocol Partner. This is
        a 16-bit read-only value. If the aggregation is manually
        configured, this Key value will be a value assigned by the
        local System."
    REFERENCE
        "6.3.1.1.12"
    ::= { dot3adAggEntry 10 }


dot3adAggCollectorMaxDelay OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The value of this 16-bit read-write attribute defines
        the maximum delay, in tens of microseconds, that
        may be imposed by the Frame Collector between
        receiving a frame from an Aggregator Parser, and
        either delivering the frame to its MAC Client or discarding
        the frame (see 5.2.3.1.1)."
    REFERENCE
        "6.3.1.1.32"
    ::= { dot3adAggEntry 11 }


-- ----------------------------------------------------------
-- The Aggregation Port List Table
-- ----------------------------------------------------------


dot3adAggPortListTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggPortListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains a list of all the ports
        associated with each Aggregator."
    REFERENCE
        "6.3.1.1.30"
    ::= { dot3adAgg 2 }


dot3adAggPortListEntry OBJECT-TYPE
    SYNTAX      Dot3adAggPortListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of the ports associated with a given Aggregator.
        This is indexed by the ifIndex of the Aggregator."
    INDEX { dot3adAggIndex }
    ::= { dot3adAggPortListTable 1 }
```

```
Dot3adAggPortListEntry ::=
    SEQUENCE {
        dot3adAggPortListPorts
            PortList
    }


dot3adAggPortListPorts OBJECT-TYPE
    SYNTAX      PortList
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The complete set of ports currently associated with
        this Aggregator.  Each bit set in this list represents
        an Actor Port member of this Link Aggregation."
    REFERENCE
        "6.3.1.1.30"
    ::= { dot3adAggPortListEntry 1 }


-- ----------------------------------------------------------
-- The Aggregation Port Table
-- ----------------------------------------------------------


dot3adAggPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains Link Aggregation Control
        configuration information about every
        Aggregation Port associated with this device.
        A row appears in this table for each physical port."
    REFERENCE
        "6.3.2"
    ::= { dot3adAggPort 1 }


dot3adAggPortEntry OBJECT-TYPE
    SYNTAX      Dot3adAggPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of Link Aggregation Control configuration
        parameters for each Aggregation Port on this device."
    INDEX      { dot3adAggPortIndex }
    ::= { dot3adAggPortTable 1 }


Dot3adAggPortEntry ::=
    SEQUENCE {
        dot3adAggPortIndex
            InterfaceIndex,
        dot3adAggPortActorSystemPriority
            INTEGER,
        dot3adAggPortActorSystemID
```



```
            MacAddress,
        dot3adAggPortActorAdminKey
            LacpKey,
        dot3adAggPortActorOperKey
            LacpKey,
        dot3adAggPortPartnerAdminSystemPriority
            INTEGER,
        dot3adAggPortPartnerOperSystemPriority
            INTEGER,
        dot3adAggPortPartnerAdminSystemID
            MacAddress,
        dot3adAggPortPartnerOperSystemID
            MacAddress,
        dot3adAggPortPartnerAdminKey
            LacpKey,
        dot3adAggPortPartnerOperKey
            LacpKey,
        dot3adAggPortSelectedAggID
            InterfaceIndex,
        dot3adAggPortAttachedAggID
            InterfaceIndex,
        dot3adAggPortActorPort
            INTEGER,
        dot3adAggPortActorPortPriority
            INTEGER,
        dot3adAggPortPartnerAdminPort
            INTEGER,
        dot3adAggPortPartnerOperPort
            INTEGER,
        dot3adAggPortPartnerAdminPortPriority
            INTEGER,
        dot3adAggPortPartnerOperPortPriority
            INTEGER,
        dot3adAggPortActorAdminState
            LacpState,
        dot3adAggPortActorOperState
            LacpState,
        dot3adAggPortPartnerAdminState
            LacpState,
        dot3adAggPortPartnerOperState
            LacpState,
        dot3adAggPortAggregateOrIndividual
            TruthValue
    }


dot3adAggPortIndex OBJECT-TYPE
    SYNTAX       InterfaceIndex
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The ifIndex of the port"
    REFERENCE
        "6.3.2.1.1"
    ::= { dot3adAggPortEntry 1 }


dot3adAggPortActorSystemPriority OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
```

```
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "A 2-octet read-write value used to define the priority
            value associated with the Actor's System ID."
        REFERENCE
            "6.3.2.1.2"
        ::= { dot3adAggPortEntry 2 }


    dot3adAggPortActorSystemID OBJECT-TYPE
        SYNTAX       MacAddress
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A 6-octet read-only MAC address value that defines
            the value of the System ID for the System that contains this
            Aggregation Port."
        REFERENCE
            "6.3.2.1.3"
        ::= { dot3adAggPortEntry 3 }


    dot3adAggPortActorAdminKey OBJECT-TYPE
        SYNTAX       LacpKey
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "The current administrative value of the Key for the
            Aggregation Port. This is a 16-bit read-write value.
            The meaning of particular Key values is of local significance."
        REFERENCE
            "6.3.2.1.4"
        ::= { dot3adAggPortEntry 4 }


    dot3adAggPortActorOperKey OBJECT-TYPE
        SYNTAX       LacpKey
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "The current operational value of the Key for the
            Aggregation Port. This is a 16-bit read-only value.
            The meaning of particular Key values is of local significance."
        REFERENCE
            "6.3.2.1.5"
        ::= { dot3adAggPortEntry 5 }


    dot3adAggPortPartnerAdminSystemPriority OBJECT-TYPE
        SYNTAX       INTEGER (0..65535)
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "A 2-octet read-write value used to define the administrative
            value of priority associated with the Partner's System ID. The
            assigned value is used, along with the value of
            aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey,
            aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority,
```



```
        in order to achieve manually configured aggregation."
    REFERENCE
        "6.3.2.1.6"
    ::= { dot3adAggPortEntry 6 }


dot3adAggPortPartnerOperSystemPriority OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 2-octet read-only value indicating the operational value
        of priority associated with the Partner's System ID. The
        value of this attribute may contain the manually configured value
        carried in aAggPortPartnerAdminSystemPriority
        if there is no protocol Partner."
    REFERENCE
        "6.3.2.1.7"
    ::= { dot3adAggPortEntry 7 }


dot3adAggPortPartnerAdminSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A 6-octet read-write MACAddress value representing
        the administrative value of the Aggregation Port's protocol
        Partner's System ID. The assigned value is used, along with
        the value of aAggPortPartnerAdminSystemPriority,
        aAggPortPartnerAdminKey, aAggPortPartnerAdminPort,
        and aAggPortPartnerAdminPortPriority, in order to
        achieve manually configured aggregation."
    REFERENCE
        "6.3.2.1.8"
    ::= { dot3adAggPortEntry 8 }


dot3adAggPortPartnerOperSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A 6-octet read-only MACAddress value representing
        the current value of the Aggregation Port's protocol Partner's
        System ID. A value of zero indicates that there is no known
        protocol Partner. The value of this attribute may contain the
        manually configured value carried in
        aAggPortPartnerAdminSystemID if there is no protocol Partner."
    REFERENCE
        "6.3.2.1.9"
    ::= { dot3adAggPortEntry 9 }


dot3adAggPortPartnerAdminKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
```

```
        "The current administrative value of the Key for the
        protocol Partner. This is a 16-bit read-write value.
        The assigned value is used, along with the value of
        aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID,
        aAggPortPartnerAdminPort, and aAggPortPartnerAdminPortPriority,
        in order to achieve manually configured aggregation.
    REFERENCE
        "6.3.2.1.10"
    ::= { dot3adAggPortEntry 10 }


dot3adAggPortPartnerOperKey OBJECT-TYPE
    SYNTAX       LacpKey
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current operational value of the Key for the
        protocol Partner. The value of this attribute may contain
        the manually configured value carried in
        aAggPortPartnerAdminKey if there is no protocol Partner.
        This is a 16-bit read-only value."
    REFERENCE
        "6.3.2.1.11"
    ::= { dot3adAggPortEntry 11 }


dot3adAggPortSelectedAggID OBJECT-TYPE
    SYNTAX       InterfaceIndex
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The identifier value of the Aggregator that this Aggregation
        Port has currently selected. Zero indicates that the Aggregation
        Port has not selected an Aggregator, either because it is in the
        process of detaching from an Aggregator or because there is no
          suitable Aggregator available for it to select. This value is read-
     only."
    REFERENCE
        "6.3.2.1.12"
    ::= { dot3adAggPortEntry 12 }


dot3adAggPortAttachedAggID OBJECT-TYPE
    SYNTAX       InterfaceIndex
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The identifier value of the Aggregator that this Aggregation
        Port is currently attached to. Zero indicates that the Aggregation
         Port is not currently attached to an Aggregator. This value is read-
     only."
    REFERENCE
        "6.3.2.1.13"
    ::= { dot3adAggPortEntry 13 }


dot3adAggPortActorPort OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-only
```



```
    STATUS       current
    DESCRIPTION
        "The port number locally assigned to the Aggregation Port.
        The port number is communicated in LACPDUs as the
        Actor_Port. This value is read-only."
    REFERENCE
        "6.3.2.1.14"
    ::= { dot3adAggPortEntry 14 }


dot3adAggPortActorPortPriority OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The priority value assigned to this Aggregation Port.
        This 16-bit value is read-write."
    REFERENCE
        "6.3.2.1.15"
    ::= { dot3adAggPortEntry 15 }


dot3adAggPortPartnerAdminPort OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The current administrative value of the port number
        for the protocol Partner. This is a 16-bit read-write value.
        The assigned value is used, along with the value of
        aAggPortPartnerAdminSystemPriority,
        aAggPortPartnerAdminSystemID, aAggPortPartnerAdminKey,
        and aAggPortPartnerAdminPortPriority,
        in order to achieve manually configured aggregation."
    REFERENCE
        "6.3.2.1.16"
    ::= { dot3adAggPortEntry 16 }


dot3adAggPortPartnerOperPort OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The operational port number assigned to this Aggregation
        Port by the Aggregation Port's protocol Partner. The value
        of this attribute may contain the manually configured value
        carried in aAggPortPartnerAdminPort if there is no protocol
        Partner. This 16-bit value is read-only."
   REFERENCE
        "6.3.2.1.17"
    ::= { dot3adAggPortEntry 17 }


dot3adAggPortPartnerAdminPortPriority OBJECT-TYPE
    SYNTAX       INTEGER (0..65535)
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
```

```
            "The current administrative value of the port priority
            for the protocol Partner. This is a 16-bit read-write value.
            The assigned value is used, along with the value of
            aAggPortPartnerAdminSystemPriority, aAggPortPartnerAdminSystemID,
            aAggPortPartnerAdminKey, and aAggPortPartnerAdminPort,
            in order to achieve manually configured aggregation."
        REFERENCE
            "6.3.2.1.18"
        ::= { dot3adAggPortEntry 18 }


    dot3adAggPortPartnerOperPortPriority OBJECT-TYPE
        SYNTAX       INTEGER (0..65535)
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The priority value assigned to this Aggregation Port
            by the Partner. The value of this attribute may contain the
            manually configured value carried in
            aAggPortPartnerAdminPortPriority if there is no
            protocol Partner. This 16-bit value is read-only."
        REFERENCE
            "6.3.2.1.19"
        ::= { dot3adAggPortEntry 19 }


    dot3adAggPortActorAdminState OBJECT-TYPE
        SYNTAX       LacpState
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "A string of 8 bits, corresponding to the administrative values
            of Actor_State (5.4.2) as transmitted by the Actor in LACPDUs.
            The first bit corresponds to bit 0 of Actor_State (LACP_Activity),
            the second bit corresponds to bit 1 (LACP_Timeout), the third bit
            corresponds to bit 2 (Aggregation), the fourth bit corresponds to
            bit 3 (Synchronization), the fifth bit corresponds to bit 4
            (Collecting), the sixth bit corresponds to bit 5 (Distributing),
            the seventh bit corresponds to bit 6 (Defaulted), and the eighth
            bit corresponds to bit 7 (Expired). These values allow
            administrative control over the values of LACP_Activity,
            LACP_Timeout and Aggregation. This attribute value is read-write."
        REFERENCE
            "6.3.2.1.20"
        ::= { dot3adAggPortEntry 20 }


    dot3adAggPortActorOperState OBJECT-TYPE
        SYNTAX       LacpState
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "A string of 8 bits, corresponding to the current
            operational values of Actor_State as transmitted by the
            Actor in LACPDUs. The bit allocations are as defined in
            6.3.2.1.20. This attribute value is read-only."
        REFERENCE
            "6.3.2.1.21"
        ::= { dot3adAggPortEntry 21 }
```



```
dot3adAggPortPartnerAdminState OBJECT-TYPE
    SYNTAX       LacpState
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A string of 8 bits, corresponding to the current administrative
        value of Actor_State for the protocol Partner. The bit
        allocations are as defined in 6.3.2.1.20. This attribute value is
        read-write. The assigned value is used in order to achieve
        manually configured aggregation."
    REFERENCE
        "6.3.2.1.22"
    ::= { dot3adAggPortEntry 22 }


dot3adAggPortPartnerOperState OBJECT-TYPE
    SYNTAX       LacpState
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A string of 8 bits, corresponding to the current values of
        Actor_State in the most recently received LACPDU transmitted
        by the protocol Partner. The bit allocations are as defined in
        6.3.2.1.20. In the absence of an active protocol Partner, this
        value may reflect the manually configured value
        aAggPortPartnerAdminState. This attribute value is read-only."
    REFERENCE
        "6.3.2.1.23"
    ::= { dot3adAggPortEntry 23 }


dot3adAggPortAggregateOrIndividual OBJECT-TYPE
    SYNTAX       TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A read-only Boolean value indicating whether the
        Aggregation Port is able to Aggregate ('TRUE') or is
        only able to operate as an Individual link ('FALSE')."
    REFERENCE
        "6.3.2.1.24"
    ::= { dot3adAggPortEntry 24 }


-- ----------------------------------------------------------
-- LACP Statistics Table
-- ----------------------------------------------------------


dot3adAggPortStatsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggPortStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains Link Aggregation information
        about every port that is associated with this device.
        A row appears in this table for each physical port."
```

```
        REFERENCE
            "6.3.3"
        ::= { dot3adAggPort 2 }


    dot3adAggPortStatsEntry OBJECT-TYPE
        SYNTAX      Dot3adAggPortStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A list of Link Aggregation Control Protocol statistics
            for each port on this device."
        INDEX { dot3adAggPortIndex }
        ::= { dot3adAggPortStatsTable 1 }


    Dot3adAggPortStatsEntry ::=
        SEQUENCE {
            dot3adAggPortStatsLACPDUsRx
                Counter32,
            dot3adAggPortStatsMarkerPDUsRx
                Counter32,
            dot3adAggPortStatsMarkerResponsePDUsRx
                Counter32,
            dot3adAggPortStatsUnknownRx
                Counter32,
            dot3adAggPortStatsIllegalRx
                Counter32,
            dot3adAggPortStatsLACPDUsTx
                Counter32,
            dot3adAggPortStatsMarkerPDUsTx
                Counter32,
            dot3adAggPortStatsMarkerResponsePDUsTx
                Counter32
        }


    dot3adAggPortStatsLACPDUsRx OBJECT-TYPE
        SYNTAX       Counter32
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The number of valid LACPDUs received on this
            Aggregation Port. This value is read-only."
        REFERENCE
            "6.3.3.1.2"
        ::= { dot3adAggPortStatsEntry 1 }


    dot3adAggPortStatsMarkerPDUsRx OBJECT-TYPE
        SYNTAX       Counter32
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The number of valid Marker PDUs received on this
            Aggregation Port. This value is read-only."
        REFERENCE
            "6.3.3.1.3"
        ::= { dot3adAggPortStatsEntry 2 }
```



```
dot3adAggPortStatsMarkerResponsePDUsRx OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of valid Marker Response PDUs received on this
        Aggregation Port. This value is read-only."
    REFERENCE
        "6.3.3.1.4"
    ::= { dot3adAggPortStatsEntry 3 }


dot3adAggPortStatsUnknownRx OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of frames received that either:
        - carry the Slow Protocols Ethernet Type value
          (IEEE Std 802.3 Annex 57A.4), but contain an
          unknown PDU, or:
        - are addressed to the Slow Protocols group MAC
          Address (IEEE Std 802.3 Annex 57A.4), but do
          not carry the Slow Protocols Ethernet Type.
        This value is read-only."
    REFERENCE
        "6.3.3.1.5"
    ::= { dot3adAggPortStatsEntry 4 }


dot3adAggPortStatsIllegalRx OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of frames received that carry the Slow
        Protocols Ethernet Type value (IEEE Std 802.3 Annex
        57A.4), but contain a badly formed PDU or an illegal
        value of Protocol Subtype (IEEE Std 802.3 Annex 57A.4).
        This value is read-only."
    REFERENCE
        "6.3.3.1.6"
    ::= { dot3adAggPortStatsEntry 5 }


dot3adAggPortStatsLACPDUsTx OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of LACPDUs transmitted on this
        Aggregation Port. This value is read-only."
    REFERENCE
        "6.3.3.1.7"
    ::= { dot3adAggPortStatsEntry 6 }
```

```
dot3adAggPortStatsMarkerPDUsTx OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of Marker PDUs transmitted on this
        Aggregation Port. This value is read-only."
    REFERENCE
        "6.3.3.1.8"
    ::= { dot3adAggPortStatsEntry 7 }


dot3adAggPortStatsMarkerResponsePDUsTx OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of Marker Response PDUs transmitted
        on this Aggregation Port. This value is read-only."
    REFERENCE
        "6.3.3.1.9"
    ::= { dot3adAggPortStatsEntry 8 }


-- ----------------------------------------------------------------
-- LACP Debug Table
-- ----------------------------------------------------------------
dot3adAggPortDebugTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggPortDebugEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains Link Aggregation debug
        information about every port that is associated with
        this device.  A row appears in this table for each
        physical port."
    REFERENCE
        "6.3.4"
    ::= { dot3adAggPort 3 }


dot3adAggPortDebugEntry OBJECT-TYPE
    SYNTAX      Dot3adAggPortDebugEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of the debug parameters for a port."
    INDEX { dot3adAggPortIndex }
    ::= { dot3adAggPortDebugTable 1 }


Dot3adAggPortDebugEntry ::=
    SEQUENCE {
        dot3adAggPortDebugRxState
            INTEGER,
        dot3adAggPortDebugLastRxTime
            TimeTicks,
        dot3adAggPortDebugMuxState
            INTEGER,
```



```
        dot3adAggPortDebugMuxReason
            DisplayString,
        dot3adAggPortDebugActorChurnState
            ChurnState,
        dot3adAggPortDebugPartnerChurnState
            ChurnState,
        dot3adAggPortDebugActorChurnCount
            Counter32,
        dot3adAggPortDebugPartnerChurnCount
            Counter32,
        dot3adAggPortDebugActorSyncTransitionCount
            Counter32,
        dot3adAggPortDebugPartnerSyncTransitionCount
            Counter32,
        dot3adAggPortDebugActorChangeCount
            Counter32,
        dot3adAggPortDebugPartnerChangeCount
            Counter32
    }


dot3adAggPortDebugRxState OBJECT-TYPE
    SYNTAX       INTEGER {
                    current(1),
                    expired(2),
                    defaulted(3),
                    initialize(4),
                    lacpDisabled(5),
                    portDisabled(6)
                 }
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This attribute holds the value 'current' if the Receive
        state machine for the Aggregation Port is in the
        CURRENT state, 'expired' if the Receive state machine
        is in the EXPIRED state, 'defaulted' if the Receive state
        machine is in the DEFAULTED state, 'initialize' if the
        Receive state machine is in the INITIALIZE state,
        'lacpDisabled' if the Receive state machine is in the
        LACP_DISABLED state, or 'portDisabled' if the Receive
        state machine is in the PORT_DISABLED state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.2"
    ::= { dot3adAggPortDebugEntry 1 }


dot3adAggPortDebugLastRxTime OBJECT-TYPE
    SYNTAX       TimeTicks
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The value of aTimeSinceSystemReset (F.2.1) when
        the last LACPDU was received by this Aggregation Port.
        This value is read-only."
    REFERENCE
        "6.3.4.1.3"
    ::= { dot3adAggPortDebugEntry 2 }
```

```
dot3adAggPortDebugMuxState OBJECT-TYPE
    SYNTAX       INTEGER {
                    detached(1),
                    waiting(2),
                    attached(3),
                    collecting(4),
                    distributing(5),
                    collecting_distributing(6)
                }
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This attribute holds the value 'detached' if the Mux
        state machine (5.4.14) for the Aggregation Port is
        in the DETACHED state, 'waiting' if the Mux state machine
        is in the WAITING state, 'attached' if the Mux state
        machine for the Aggregation Port is in the ATTACHED
        state, 'collecting' if the Mux state machine for the
        Aggregation Port is in the COLLECTING state, 'distributing'
        if the Mux state machine for the Aggregation Port is
        in the DISTRIBUTING state, and 'collecting_distributing'
        if the Mux state machine for the Aggregation Port is in
        the COLLECTING_DISTRIBUTING state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.4"
    ::= { dot3adAggPortDebugEntry 3 }


dot3adAggPortDebugMuxReason OBJECT-TYPE
    SYNTAX       DisplayString
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A human-readable text string indicating the reason
        for the most recent change of Mux machine state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.5"
    ::= { dot3adAggPortDebugEntry 4 }


dot3adAggPortDebugActorChurnState OBJECT-TYPE
    SYNTAX       ChurnState
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The state of the Actor Churn Detection machine
        (5.4.17) for the Aggregation Port. A value of 'noChurn'
        indicates that the state machine is in either the
        NO_ACTOR_CHURN or the ACTOR_CHURN_MONITOR
        state, and 'churn' indicates that the state machine is in the
        ACTOR_CHURN state. This value is read-only."
    REFERENCE
        "6.3.4.1.6"
    ::= { dot3adAggPortDebugEntry 5 }
```



```
dot3adAggPortDebugPartnerChurnState OBJECT-TYPE
    SYNTAX       ChurnState
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The state of the Partner Churn Detection machine
        (5.4.17) for the Aggregation Port. A value of 'noChurn'
        indicates that the state machine is in either the
        NO_PARTNER_CHURN or the PARTNER_CHURN_MONITOR
        state, and 'churn' indicates that the state machine is
        in the PARTNER_CHURN state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.7"
    ::= { dot3adAggPortDebugEntry 6 }


dot3adAggPortDebugActorChurnCount OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Count of the number of times the Actor Churn state
        machine has entered the ACTOR_CHURN state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.8"
    ::= { dot3adAggPortDebugEntry 7 }


dot3adAggPortDebugPartnerChurnCount OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Count of the number of times the Partner Churn
        state machine has entered the PARTNER_CHURN state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.9"
    ::= { dot3adAggPortDebugEntry 8 }


dot3adAggPortDebugActorSyncTransitionCount OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Count of the number of times the Actor's Mux state
        machine (5.4.15) has entered the IN_SYNC state.
        This value is read-only."
    REFERENCE
        "6.3.4.1.10"
    ::= { dot3adAggPortDebugEntry 9 }


dot3adAggPortDebugPartnerSyncTransitionCount OBJECT-TYPE
    SYNTAX       Counter32
```

```
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "Count of the number of times the Partner's Mux
            state machine (5.4.15) has entered the IN_SYNC state.
            This value is read-only."
        REFERENCE
            "6.3.4.1.11"
        ::= { dot3adAggPortDebugEntry 10 }


    dot3adAggPortDebugActorChangeCount OBJECT-TYPE
        SYNTAX       Counter32
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "Count of the number of times the Actor's perception of
            the LAG ID for this Aggregation Port has changed.
            This value is read-only."
        REFERENCE
            "6.3.4.1.12"
        ::= { dot3adAggPortDebugEntry 11 }


    dot3adAggPortDebugPartnerChangeCount OBJECT-TYPE
        SYNTAX       Counter32
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "Count of the number of times the Partner's perception of
            the LAG ID (see 5.3.6.1) for this Aggregation Port has changed.
            This value is read-only."
        REFERENCE
            "6.3.4.1.13"
        ::= { dot3adAggPortDebugEntry 12 }

    -- ----------------------------------------------------------
    -- IEEE 802.3ad MIB - Conformance Information
    -- ----------------------------------------------------------

    dot3adAggConformance OBJECT IDENTIFIER ::= { lagMIB 2 }


    dot3adAggGroups OBJECT IDENTIFIER ::= { dot3adAggConformance 1 }


    dot3adAggCompliances OBJECT IDENTIFIER
        ::= { dot3adAggConformance 2 }

    -- ----------------------------------------------------------
    -- units of conformance
    -- ----------------------------------------------------------

    dot3adAggGroup OBJECT-GROUP
        OBJECTS {
            dot3adAggActorSystemID,
            dot3adAggActorSystemPriority,
            dot3adAggAggregateOrIndividual,
            dot3adAggActorAdminKey,
```



```
        dot3adAggMACAddress,
        dot3adAggActorOperKey,
        dot3adAggPartnerSystemID,
        dot3adAggPartnerSystemPriority,
        dot3adAggPartnerOperKey,
        dot3adAggCollectorMaxDelay
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing information about an
        aggregation."
    ::= { dot3adAggGroups 1 }


dot3adAggPortListGroup OBJECT-GROUP
    OBJECTS {
        dot3adAggPortListPorts
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing information about every
        port in an aggregation."
    ::= { dot3adAggGroups 2 }


dot3adAggPortGroup OBJECT-GROUP
    OBJECTS {
        dot3adAggPortActorSystemPriority,
        dot3adAggPortActorSystemID,
        dot3adAggPortActorAdminKey,
        dot3adAggPortActorOperKey,
        dot3adAggPortPartnerAdminSystemPriority,
        dot3adAggPortPartnerOperSystemPriority,
        dot3adAggPortPartnerAdminSystemID,
        dot3adAggPortPartnerOperSystemID,
        dot3adAggPortPartnerAdminKey,
        dot3adAggPortPartnerOperKey,
        dot3adAggPortSelectedAggID,
        dot3adAggPortAttachedAggID,
        dot3adAggPortActorPort,
        dot3adAggPortActorPortPriority,
        dot3adAggPortPartnerAdminPort,
        dot3adAggPortPartnerOperPort,
        dot3adAggPortPartnerAdminPortPriority,
        dot3adAggPortPartnerOperPortPriority,
        dot3adAggPortActorAdminState,
        dot3adAggPortActorOperState,
        dot3adAggPortPartnerAdminState,
        dot3adAggPortPartnerOperState,
        dot3adAggPortAggregateOrIndividual
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing information about every
        port in an aggregation."
    ::= { dot3adAggGroups 3 }


dot3adAggPortStatsGroup OBJECT-GROUP
```

```
    OBJECTS {
        dot3adAggPortStatsLACPDUsRx,
        dot3adAggPortStatsMarkerPDUsRx,
        dot3adAggPortStatsMarkerResponsePDUsRx,
        dot3adAggPortStatsUnknownRx,
        dot3adAggPortStatsIllegalRx,
        dot3adAggPortStatsLACPDUsTx,
        dot3adAggPortStatsMarkerPDUsTx,
        dot3adAggPortStatsMarkerResponsePDUsTx
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing information about every
        port in an aggregation."
    ::= { dot3adAggGroups 4 }


dot3adAggPortDebugGroup OBJECT-GROUP
    OBJECTS {
        dot3adAggPortDebugRxState,
        dot3adAggPortDebugLastRxTime,
        dot3adAggPortDebugMuxState,
        dot3adAggPortDebugMuxReason,
        dot3adAggPortDebugActorChurnState,
        dot3adAggPortDebugPartnerChurnState,
        dot3adAggPortDebugActorChurnCount,
        dot3adAggPortDebugPartnerChurnCount,
        dot3adAggPortDebugActorSyncTransitionCount,
        dot3adAggPortDebugPartnerSyncTransitionCount,
        dot3adAggPortDebugActorChangeCount,
        dot3adAggPortDebugPartnerChangeCount
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing debug information about
        every aggregated port."
    ::= { dot3adAggGroups 5 }

dot3adTablesLastChangedGroup OBJECT-GROUP
    OBJECTS {
        dot3adTablesLastChanged
    }
    STATUS current
    DESCRIPTION
        "A collection of objects providing information about the time
        of changes to the configuration of aggregations and their ports."
::= { dot3adAggGroup 6 }

-- ----------------------------------------------------------
-- compliance statements
-- ----------------------------------------------------------


dot3adAggCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
        "The compliance statement for device support of
        Link Aggregation."
```



```
    MODULE
        MANDATORY-GROUPS {
            dot3adAggGroup,
            dot3adAggPortGroup,
           dot3adTablesLastChangedGroup
        }

        GROUP       dot3adAggPortListGroup
        DESCRIPTION
            "This group is optional."

        GROUP       dot3adAggPortStatsGroup
        DESCRIPTION
            "This group is optional."

        GROUP       dot3adAggPortDebugGroup
        DESCRIPTION
            "This group is optional."

    ::= { dot3adAggCompliances 1 }

END
```