INTERNATIONAL STANDARD

ISO/IEC/IEEE 8802-1AX

First edition
2016-01-15

Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements

Part 1AX: Link Aggregation

Technologies de l'information — Télécommunications et échange d'information entre systèmes — Réseaux locaux et métropolitains — Exigences spécifiques —

Partie 1AX: Agrégation de lien

Reference number
ISO/IEC/IEEE 8802-1AX:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org
Published in Switzerland

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York
NY 10016-5997, USA
E-mail stds.ipr@ieee.org
Web www.ieee.org

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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

The main task of ISO/IEC JTC 1 is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard 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 ISO or the IEEE Standards Association.

ISO/IEC/IEEE 8802-1AX was prepared by the LAN/MAN of the IEEE Computer Society (as IEEE 8802-1AX-2014). It was adopted by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in parallel with its approval by the ISO/IEC national bodies, under the “fast-track procedure” defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE. IEEE is responsible for the maintenance of this document with participation and input from ISO/IEC national bodies.

IEEE STANDARDS ASSOCIATION IEEE

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

IEEE Std 802.1AX™-2014
(Revision of
IEEE Std 802.1AX-2008)

IEEE Std 802.1AX™-2014
(Revision of
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 10 December 2014
IEEE SA-Standards Board

Abstract: MAC-independent Link Aggregation capability and general information relevant to specific MAC types are defined in this standard. Link Aggregation allows parallel full-duplex point-to-point links to be used as if they were a single link and also supports the use of multiple links as a resilient load sharing interconnect between multiple nodes in two separately administered networks.

Keywords: Aggregated Link, Aggregator, Distributed Resilient Network Interconnect, DRNI, IEEE 802®, IEEE 802.1AX™, interconnect, Link Aggregation, Link Aggregation Group, local area network, management, Network-Network Interface, NNI

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

Print: ISBN 978-0-7381-9448-6 STD20052
PDF: ISBN 978-0-7381-9449-3 STDPD20052

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, 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.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. 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.

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

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of 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 of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive 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 comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

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

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document 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

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents 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.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, 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 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-SA Website at http://ieeexplore.ieee.org/expel/standards.jsp or contact IEEE at the address listed previously. For more information about the IEEE-SA or IOWA's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions 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 standard 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 Higher Layer LAN Protocols Working Group at the beginning of the Working Group ballot. Individuals may have not voted, voted for approval, disapproval, or abstained on this standard.

Glenn Parsons, *Working Group Chair*
John Messenger, *Working Group Vice Chair*
Stephen Haddock, *Chair, Interworking Task Group*
Michael Seaman, *Chair, Security Task Group*
Michael Johas Teener, *Chair, Time Sensitive Networking Task Group*
Pat Thaler, *Chair, Data Center Bridging Task Group*
Maximilian Riegel, *Chair, OmniRAN Task Group*
Eric Gray, *Recording Secretary*

Ting Ao
Christian Boiger
Paul Bottorff
David Chen
Feng Chen
Weiying Cheng
Diego Crupnicoff
Rodney Cummings
Patrick Diamond
Aboubacar Kader Diarra
Janos Farkas
Norman Finn
Geoffrey Garner
Anoop Ghanwani
Mark Gravel
Craig Gunther
Hitosh Hayakawa
Jeremy Hitt
Rahil Hussain
Anthony Jeffree
Peter Jones
Hal Keen
Marcel Kiessling
Yongbum Kim
Philippe Klein
Jouni Korhonen
Jeff Lynch
Ben Mack-Crane
Christophe Mangin
James McIntosh
Eric Multanen
Donald Pannell
Karen Randall
Dan Romascanu
Jessy V. Rouyer
Panagiotis Saltsidis
Behcet Sarikaya
Daniel Sexton
Johannes Specht
Kevin B. Stanton
Wilfried Steiner
Vahid Tabatabaee
Jeremy Touve
Karl Weber
Yuehua Wei
Brian Weis
Jordon Woods
Juan-Carlos Zuniga

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

Thomas Alexander
Richard Alfvin
Butch Anton
Jacob Ben Ary
Nancy Bravin
William Byrd
Juan Carreon
Keith Chow
Charles Cook
Patrick Diamond
Yezid Donoso
Sourav Dutta
Donald Eastlake, 3rd
Richard Edgar
Andrew Fieldsend
Yukihiro Fujimoto
Devon Gayle
Anoop Ghanwani
Randall Groves
Chris Guy
Stephen Haddock
Werner Hoelzl
Rita Horner
Victor Hou
Noriyuki Ikeuchi
James Innis
Osamu Ishida
Akio Iso
Atsushi Ito
Raj Jain
Anthony Jeffree
Michael Johas Teener
Peter Jones
Shinkyo Kaku
Piotr Karocki
Stuart Kerry
Yongbum Kim
Bruce Kraemer
Geoff Ladwig
Mark Laubach
John Lemon
Ru Lin
Elvis Maculuba
Roger Marks
Jeffery Masters
Brett Mcclellan
Jonathon Mclendon
Richard Mellitz
John Messenger
Charles Moorwood
Jose Morales
Michael Newman
Nick S.A. Nikjoo
Paul Nikolich
Satoshi Obara
Maximilian Riegel
Dan Romascanu
Jessy V. Rouyer
Panagiotis Saltsidis
Peter Saunderson
Michael Seaman
Kapil Sood
Thomas Starai
Rene Struik
Walter Struppler
William Taylor
Dmitri Varsanofiev
Prabodh Varshney
George Vlantis
Hung-Yu Wei
Andreas Wolf
Chun Yu
Charles Wong
Michael Wright
Oren Yuen
Daidi Zhong

When the IEEE-SA Standards Board approved this standard on 10 December 2014, it had the following membership:

John Kulick, *Chair*
Jon Walter Rosdahl, *Vice Chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Peter Balma
Farooq Bari
Ted Burse
Clint Chaplain
Stephen Dukes
Jean-Phillippe Faure
Gary Hoffman
Michael Janezic
Jeffrey Katz
Joseph L. Koepfinger*
David J. Law
Hung Ling
Oleg Logvinov
T. W. Olsen
Glenn Parsons
Ron Peterson
Adrian Stephens
Peter Sutherland
Yatin Trivedi
Phil Winston
Don Wright
Yu Yuan

*Member Emeritus

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

Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Michelle Turner
IEEE-SA Content Publishing

Kathryn Bennett
IEEE-SA Technical Community Programs

Introduction

This introduction is not part of IEEE Std 801.AX™-2014, IEEE Standard for Local and metropolitan area networks—Link Aggregation.

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

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

Contents

IEEE Standard for
Local and metropolitan area networks—

Link Aggregation

1. Overview

1.1 Scope

Link Aggregation provides protocols, procedures, and managed objects that allow the following:

- One or more parallel instances of full-duplex point-to-point links to be aggregated together to form a Link Aggregation Group (LAG), such that a MAC Client can treat the LAG as if it were a single link.
- A resilient interconnect using multiple full-duplex point-to-point links among one to three nodes in a network and one to three nodes in another, separately administered, network, along with a means to ensure that frames belonging to any given service will use the same physical path in both directions between the two networks.

This standard defines the MAC-independent Link Aggregation capability and general information relevant to specific MAC types that support Link Aggregation. The capabilities defined are compatible with previous versions of this standard.

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, and the use of multiple systems at each end of the aggregation. This allows improved utilization of available links in bridged local area network (LAN) environments, along with improved resilience in the face of failure of individual links or systems. In

applications connecting separately administered networks, the networks are isolated from each other's fault recovery events.

1.3 State diagram conventions

This document uses the state diagram conventions of IEEE Std 802.1Q™-2011, Annex E.[1]

Should a conflict exist between a state diagram and either the corresponding global transition tables or the textual description associated with the state machine, the state diagram takes precedence.

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

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®, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.[2, 3]

IEEE Std 802.1AC™, IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Service Definition.

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

IEEE Std 802.1Q™-2011, IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks.

IEEE Std 802.3™-2012, IEEE Standard for Ethernet.

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.[4]

IETF RFC 1321, The MD5 Message-Digest Algorithm, R. Rivest. April 1992.

IETF RFC 2578 (STD 58), Structure of Management Information Version 2 (SMIv2), K. McCloghrie, D. Perkins, J. Schoenwaelder. April 1999.

IETF RFC 2579 (STD 58), Textual Conventions for SMIv2, McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, April 1999.

IETF RFC 2580 (STD 58), Conformance Statements for SMIv2, McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, April 1999.

IETF RFC 2863, The Interfaces Group MIB, K. McCloghrie, F. Kastenholz, June 2000.

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

IETF RFC 3414 (STD 62), User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3), U. Blumenthal, B. Wijnen. December 2002.

IETF RFC 3415 (STD 62), View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), B. Wijnen, R. Presuhn, K. McCloghrie, December 2002.

ISO/IEC 10165-4:1992, Information technology—Open Systems Interconnection—Structure of management information—Part 4: Guidelines for the definition of managed objects.[5]

[2]IEEE publications are available from The Institute of Electrical and Electronics Engineers (http://standards.ieee.org/).

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

[4]IETF documents (i.e., RFCs) are available for download at http://www.rfc-archive.org/.

[5]ISO/IEC publications are available from the ISO Central Secretariat (http://www.iso.org/). ISO publications are also available in the United States from the American National Standards Institute (http://www.ansi.org/).

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[6]

Actor: The local entity in a Link Aggregation Control Protocol exchange.

agent: A term used to refer to the managed nodes in a network. Managed nodes are those nodes that contain a network management entity, 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 a network management entity and network management application 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.

Aggregation Key: A parameter associated with each Aggregation Port and with each Aggregator of an Aggregation System identifying those Aggregation Ports that can be aggregated together. Aggregation Ports in an Aggregation System that share the same Aggregation Key value are potentially able to aggregate together.

Aggregation Link: A LAN connecting a pair of Aggregation Systems.

Aggregation Port: A Service Access Point (SAP) in an Aggregation System that is supported by a single MAC entity.

Aggregation System: A uniquely identifiable entity comprising (among other things) an arbitrary grouping of one or more Aggregation 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.

Aggregator: A logical MAC, bound to one or more Aggregation Ports, through which the Aggregator Client is provided access to the physical media.

Aggregator Client: The layered entity immediately above the Link Aggregation Sublayer, for which the Link Aggregation sublayer provides an instance of the Internal Sublayer Service (ISS).

Aggregator Port: A Service Access Point (SAP) in an Aggregation System that is supported by an Aggregator.

bridge: A layer 2 interconnection device that conforms to IEEE Std 802.1D and/or IEEE Std 802.1Q.

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.

Conversation ID: An identifier using values in the range of 0 through 4095 to identify conversations.

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

[6] *IEEE Standards Dictionary Online* subscription is available at: http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

Distributed Relay: A functional entity, distributed over a Portal by a DR Function in each of the Systems comprising a Portal, that distributes Down frames from Gateways to Aggregators, and distributes Up frames from Aggregators to Gateways.

Distributed Resilient Network Interconnect (DRNI): Link Aggregation expanded to include at least one end of the Link Aggregation Group attached to a Portal such that independence and isolation is provided between the interconnected network segments.

NOTE—Link Aggregation as specified in Clause 6.[7]

Down frame: A frame entering a Portal through a Gateway.

DR Function: That part of a Distributed Relay residing within a single Portal System.

end station: A System attached to a local area network (LAN) that is an initial source or a final destination of frames transmitted across that LAN.

frame: A unit of data transmission on a local area network (LAN) that conveys a MAC Protocol Data Unit (MPDU) and can cause a service indication with, at a minimum, a Destination Address, Source Address, a MAC Service Data Unit (MSDU), and priority, or an MPDU that is the result of a service request with those parameters.

full duplex: A mode of operation of a network, data terminal equipment (DTE), or Medium Attachment Unit (MAU) that supports duplex transmission as defined in the *IEEE Standards Dictionary Online*.

NOTE—Within the scope of this standard, this mode of 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.)

Gateway: A connection, always virtual (not a physical link between Systems) connecting a Distributed Relay to a System, consisting of a Gateway Link and two Gateway Ports.

Gateway Conversation ID: The Conversation ID value that is used within the Distributed Relay to identify frames passing through a Gateway.

Gateway Vector: The operational Boolean vector, indexed by Gateway Conversation ID, indicating whether the indexed Gateway Conversation ID is enabled to pass through a Portal System's Gateway (FALSE = blocked). There is one Gateway Vector per Portal System in a Portal.

half duplex: A mode of operation of an CSMA/CD local area network (LAN) in which data terminal equipment (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 Ethernet protocol. (See IEEE Std 802.3.)

Internal Sublayer Service (ISS): An augmented version of the MAC Service, defined in IEEE Std 802.1AC.

Intra-Portal Link (IPL): A logical link used to connect the Distributed Relay (DR) Functions comprising a Distributed Relay.

Intra-Portal Port (IPP): A port to an Intra-Portal Link.

[7]Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

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

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

Link Aggregation Group (LAG): A group of links that appear to an Aggregator Client as if they were a single link. A LAG can connect two Aggregation Systems, an Aggregation System and a Portal, or two Portals. One or more conversations may be associated with each link that is part of a LAG.

Long LACPDU: A Version 2 or higher version Link Aggregation Control Protocol Data Unit (LACPDU) of a frame size that is larger than 128 octets.

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

MAC: The data link sublayer that is responsible for transferring data to and from the Physical Layer.

Partner: The remote entity in a Link Aggregation Control Protocol exchange.

Port Conversation ID: The Conversation ID value that is used to select frames passing through an Aggregation Port.

Portal: One end of a Distributed Resilient Network Interconnect (DRNI); one to three Systems, each with physical links that together comprise a Link Aggregation Group. The Portal's Systems cooperate to emulate the presence of a single Aggregation System to which the entire Link Aggregation Group is attached.

Portal System: A System that participates in a Portal.

Portal System Number: An integer from 1 through 3, inclusive, uniquely identifying a Portal System within its Portal.

selection algorithm: The algorithm used to assign frames to Conversation IDs and Conversation IDs to Aggregation Ports and, in the context of Distributed Resilient Network Interconnect (DRNI), also to Gateways.

Service ID: A value extracted from the header of a frame (VID, I-SID, etc.), received from an Aggregation Link, that identifies the service instance with which that frame is associated.

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

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

Type/Length/Value (TLV): A 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 defined protocol.

Up frame: A frame entering a Portal through an Aggregation Port.

4. Acronyms and abbreviations

This standard contains the following acronyms and abbreviations:

ANSI	American National Standards Institute
CID	Company ID[8] (see also OUI)
DA	destination address
DRCP	Distributed Relay Control Protocol
DRCPDU	Distributed Relay Control Protocol Data Unit
DRNI	Distributed Resilient Network Interconnect
DTE	data terminal equipment
FCS	frame check sequence
IEC	International Electrotechnical Commission
IPL	Intra-Portal Link
IPP	Intra-Portal Port
ISO	International Organization for Standardization
ISS	Internal Sublayer Service
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
LLDP	Link Layer Discovery Protocol
LLID	logical link identifier
MAC	Medium Access Control
MAN	metropolitan area network
MIB	management information base
NTT	Need To Transmit
OUI	organizationally unique identifier
PDU	Protocol Data Unit
PICS	protocol implementation conformance statement
PSI	Portal System Isolated
SA	source address
TLV	Type/Length/Value
TPMR	Two-Port Media Access Control (MAC) Relay
UCT	unconditional transition

[8] See http://standards.ieee.org/develop/regauth/tut/eui.pdf.

5. Conformance

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

5.1 Requirements terminology

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

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

The protocol implementation conformance statement (PICS) proformas (see Annex A) reflect the occurrences of the words "shall," "may," and "should" within the standard.

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

In this clause the term ***conformant System*** refers to a System that, for all ports for which support is claimed, conforms to the provisions of this standard for Link Aggregation (5.3) or Distributed Resilient Network Interconnect (DRNI)(5.4).

5.2 Protocol Implementation Conformance Statement

The supplier of an implementation that is claimed to conform to this standard shall provide the information necessary to identify both the supplier and the implementation, and shall complete a copy of the PICS proforma provided in Annex A.

5.3 Link Aggregation requirements

A conformant System, shall:

a) Support the Link Aggregation Sublayer and conform to the state machines and procedures in 6.2;
b) Support the Link Aggregation Control Protocol (LACP) and conform to the state machines and procedures in 6.3 and 6.4;
c) Transmit and receive version 1 Link Aggregation Control Protocol Data Unit (LACPDUs) in the formats specified in 6.4.2;
d) Implement the Marker Responder as specified in 6.5.4.2 and respond to received Marker PDUs as specified in 6.5.1.

5.3.1 Link Aggregation options

A conformant System, may:

a) Support the Marker Protocol and conform to the state machines and procedures in 6.5 and transmit and receive required Marker and Marker Response PDUs in the formats specified in 6.5.3;
b) Support the management functionality for Link Aggregation as specified in Clause 7;
c) Support SMIv2 management information base (MIB) modules for the management of Link Aggregation capabilities (Annex D);
d) Support Aggregation Port Debug Information package support as specified in 7.3 and in this case shall support the Churn Detection machine as specified in 6.4.17;
e) Conform to the required specifications of the Link Layer Discovery Protocol (LLDP) of IEEE Std 802.1AB. A conformant System that supports LLDP shall provide an LLDP Parser/Multiplexer to attach zero or more instances of LLDP to each Aggregation Port as specified in 6.1.3;
f) Implement the Protocol Parser/Multiplexer (6.2.7) for a protocol not explicitly specified in 6.2.7;
g) Support Per-service frame distribution as specified in 8.2. A conformant System that supports Per-service frame distribution shall support:
    1) Conversation-sensitive frame collection and distribution state diagrams as specified in 6.6.1; and
    2) Classification by Customer VLAN Tag for C-tagged interfaces and classification by Service VLAN tag for S-tagged and B-tagged interfaces,

    and may support:
    3) Conversation-sensitive LACP operation as specified in 6.6.2;
    4) Classification by Backbone Service Instance Tag as specified in 8.2.2;
    5) Classification by other methods not specified by this standard;
h) Transmit and receive version 2 LACPDUs in the formats specified in 6.4.2. A conformant System that supports version 2 LACPDUs shall support the Long LACPDU machine as specified in 6.4.18.

5.4 Distributed Resilient Network Interconnect requirements

A conformant System, shall:

a) Conform to the requirements specified for Link Aggregation as specified in 5.3;
b) Support a Portal consisting of two Portal Systems as specified in Clause 9. A conformant System that supports a Portal consisting of two Portal Systems shall:
    1) Support a DR Function having a single Intra-Portal Link (IPP) and conform to the state machines and procedures in 9.2 and 9.3 that are applicable to a Portal System supporting a single IPP;
    2) Support the Distributed Relay Control Protocol (DRCP) and conform to the state machine and procedures applicable to a Portal System supporting a single IPP as specified in 9.4;
    3) Transmit and receive required Distributed Relay Control Protocol Data Units (DRCPDUs) in the formats specified in 9.4.3 that are applicable to a Portal consisting of two Portal Systems;
c) Support using separate physical links for the Intra-Portal Link (IPL) and the network links as specified in 9.3.2.

5.4.1 Distribution Resilient Network Interconnect options

A conformant System, may implement any of the options specified for Link Aggregation (5.3.1), and may:

a) Support a Portal consisting of three Portal Systems as specified in Clause 9. A conformant System that supports a Portal consisting of three Portal Systems shall:

1) Support a DR Function having two IPPs and conform to the state machines and procedures in 9.2 and 9.3 that are applicable to a Portal System supporting two IPPs;
2) Support the DRCP and conform to the state machine and procedures applicable to a Portal System supporting two IPPs as specified in 9.4;
3) Transmit and receive required DRCPDUs in the formats specified in 9.4.3 for three Portal Systems;

b) Support any of the methods in 9.3.2, by which the Systems can distinguish frames on a network link from frames on a particular IPL. A System that supports Network / IPL sharing by time (9.3.2.1) may also support MAC address synchronization as specified in Annex G.

c) Transmit and receive DRCPDUs in the formats specified in item b3) in , including any additional DRCP Type/Length/Values (TLVs) from Table 9-8.

6. Link Aggregation

6.1 Overview

This clause defines an optional Link Aggregation sublayer for use with links offering the IEEE 802.1AC Internal Sublayer Service (ISS). Link Aggregation allows one or more links to be aggregated together to form a Link Aggregation Group (LAG), such that an Aggregator Client can treat the LAG as if it were a single link. To this end, Link Aggregation specifies the establishment of data terminal equipment (DTE)-to-DTE logical links, consisting of N parallel instances of full-duplex point-to-point links.

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

6.1.1 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 LAG need not cause failure from the perspective of an Aggregator Client.
d) **Load sharing**—Aggregator Client traffic may be distributed across multiple links.
e) **Automatic configuration**—In the absence of manual overrides, an appropriate set of LAGs 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 milliseconds for link down events and 1 s or less for link up events.

NOTE 1—Timing out LACPDU exchanges is the method of last resort for detecting link failures and shifting data flows to other physical links. The MAC_Operational status generated by the link hardware is the first choice for link failure detection. IEEE Std 802.1Q-2014, Clause 18, provides a method for detecting link failures (also indicated via the MAC_Operational status) when hardware means are not applicable.

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 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 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 frame formats**—Link aggregation neither adds to, nor changes the contents of, frames exchanged between Aggregator Clients.

m) **Network management support**—The standard specifies appropriate management objects for configuration, monitoring, and control of Link Aggregation.
n) **Dissimilar MACs**—Link Aggregation can be used over any physical or logical medium offering the ISS.

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

o) **Multipoint Aggregations**—The mechanisms specified in this clause do not support aggregations among more than two Systems. (See Clause 9 for aggregations among two Portals each consisting of up to three Systems.)
p) **Half-duplex operation**—Link Aggregation is supported only on point-to-point links with MACs operating in full-duplex mode.

Aggregation of links of different data rates is not prohibited nor required by this standard. Determining how to distribute traffic across links of different data rates is beyond the scope of this standard.

NOTE 2—Previous versions of this standard restricted the data rate of the aggregated links. However, the specified mechanisms did not depend on whether the links operated at the same rate or not.

6.1.2 Positioning of Link Aggregation within the IEEE 802 architecture

Link Aggregation comprises an optional sublayer between an Aggregator Client and the MAC. Figure 6-1 depicts the positioning of the Link Aggregation sublayer in the IEEE 802 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 Aggregator Client. (See also IEEE Std 802.1Q-2014, Figure 22-8.)

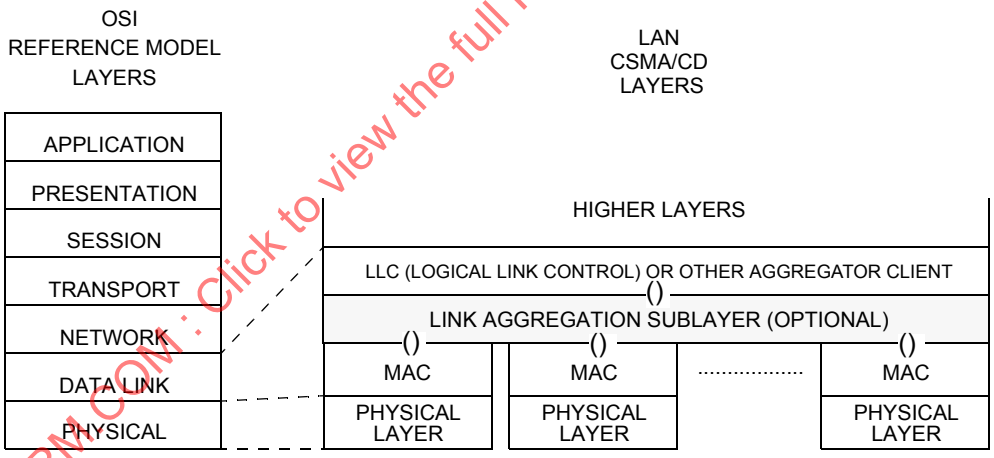


Figure 6-1—Architectural positioning of Link Aggregation sublayer

Figure 6-2 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.

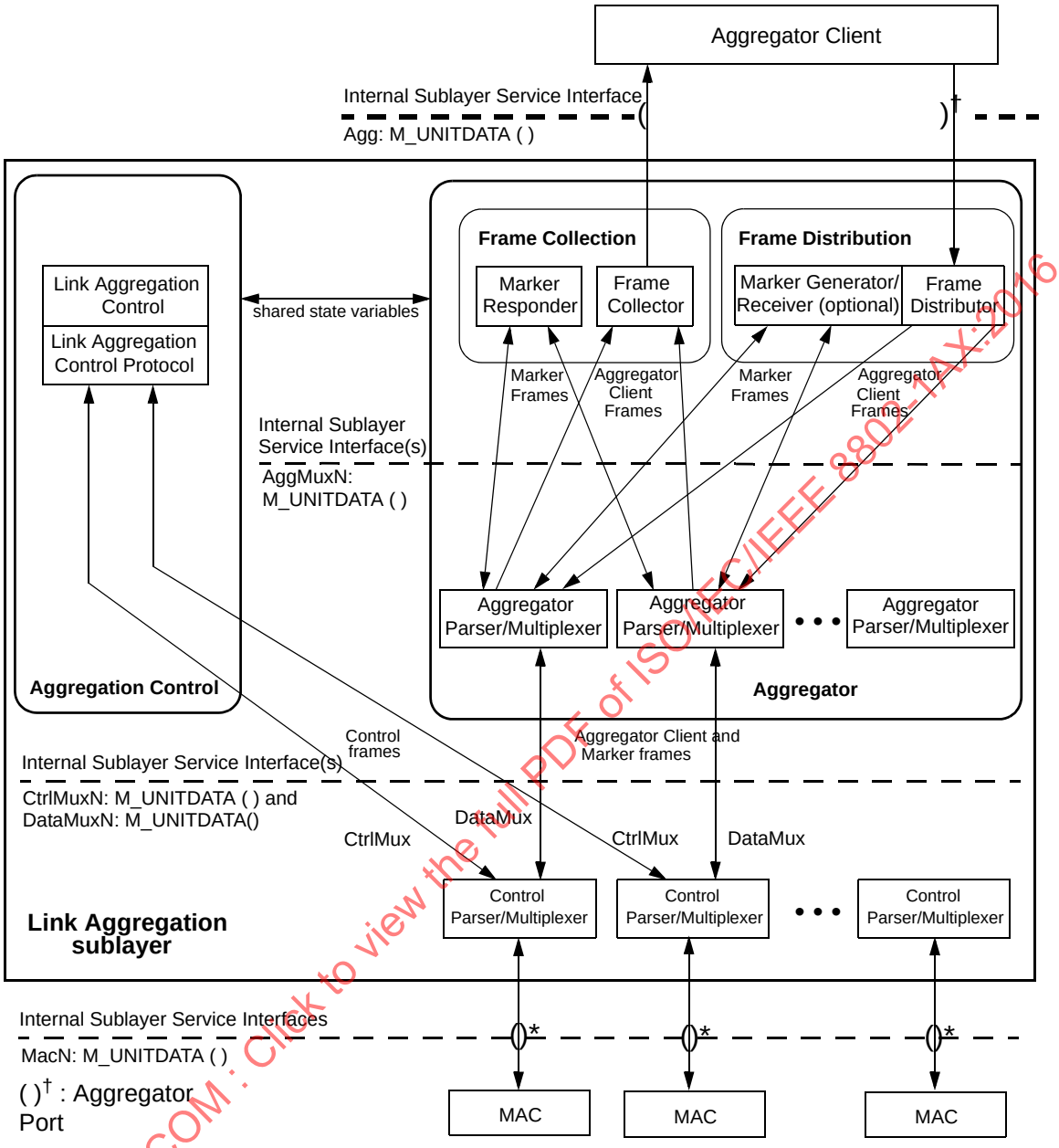


Figure 6-2—Link Aggregation sublayer block diagram

NOTE—Earlier editions of IEEE Std 802.1AX and IEEE Std 802.3, Clause 43, used the term “MAC Client” instead of the term “Aggregator Client,” and placed Link Aggregation between the IEEE 802.3 MAC sublayer and the IEEE 802.3 MAC Control sublayer. This standard repositions Link Aggregation immediately above the MAC sublayer. This rearrangement of the abstract layering diagram is intended to align the standard with common industry practice; it does not cause an implementation that is conformant to earlier editions of the standard to become non-conformant with this edition.

6.1.3 LLDP Parser/Multiplexer

Not shown in Figure 6-1 is the fact that some protocols require that the higher layers be able to access physical ports directly, without passing through the Link Aggregation sublayer. Some protocols specify a

means whereby this access is accomplished. See, for example, IEEE Std 802.1X™-2010, Figure 6-2, the "SecY."

In order to satisfy the needs of other protocols, the Protocol Parser/Multiplexer (6.2.7) is provided. In particular, if IEEE Std 802.1AB™ is implemented in a System supporting Link Aggregation, an LLDP Parser/Multiplexer shall be provided to attach zero or more instances of LLDP to each Aggregation Port as configured by the system administrator.

There are N LLDP Parser/Multiplexers, one for each Aggregation Port. Each LLDP Parser/Multiplexer is an instance of the Protocol Parser/Multiplexer described in 6.2.7 and is inserted as a shim layer, either between the Control Parser/Multiplexer and the MAC layer, or between the Aggregator Parser/Multiplexer and the Control Parser/Multiplexer, at the discretion of the implementer. Specifically:

a) The *DownPort* of Protocol Parser/Multiplexer N is either the *DataMuxN* of Control Parser/Multiplexer N (6.2.10) or the *MacN* of MAC N, as determined by the position of the LLDP Parser/Multiplexer in the protocol stack.
b) The *ControlPort* of Protocol Parser/Multiplexer N is utilized by the LLDP instances.
c) The *DataPort* of Protocol Parser/Multiplexer N is either the *DataMuxN* of Aggregator Parser/Multiplexer N (6.2.8) or the *MacN* of Control Parser/Multiplexer N, as determined by the position of the LLDP Parser/Multiplexer in the protocol stack.
d) The IsControlFrame function is defined in 6.1.3.1.1.

NOTE—Any functional entity that operates above the Aggregator is under the control and the specification of the System implementing Link Aggregation (in Bridges, for example, this is provided by the Transmit and Receive functions specified in 8.5 of IEEE Std 802.1Q-2014). The discussion of the position of any Protocol Parser/Multiplexer entity that is above the Aggregator is not in scope of this standard.

6.1.3.1 LLDP Parser state diagram

6.1.3.1.1 LLDP Parser Function

IsControlFrame
Returns Boolean value: (Length/Type == LLDP_Type && DA == LLDP_Multicast address)

6.1.3.1.2 Constants

LLDP_Multicast address
The value of the LLDP DA field. (See IEEE Std 802.1AB-2009, 7.1.)
LLDP_Type
The value of the LLDP Length/Type field. (See IEEE Std 802.1AB-2009, Figure C.1.)

6.1.3.1.3 Variables

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

6.1.3.1.4 State diagram

The LLDP Parser state diagram, when supported, shall implement the function specified in Figure 6-5 with its associated parameters as specified is 6.1.3.

6.2 Link Aggregation operation

As depicted in Figure 6-2, the Link Aggregation sublayer comprises the following functions:

a) *Frame Distribution.* This block is responsible for taking frames submitted by the Aggregator Client and submitting them for transmission on the appropriate Aggregation Port, based on a frame distribution algorithm employed by the Frame Distributor. Frame Distribution includes a *Frame Distributor* and an optional *Marker Generator/Receiver* used for the Marker protocol. (See 6.2.4, 6.2.5, and 6.5.)
b) *Frame Collection.* This block is responsible for passing frames received from the various Aggregation Ports to the Aggregator Client. Frame Collection includes a *Frame Collector* and a *Marker Responder,* used for the Marker protocol. (See 6.2.3, 6.2.6, and 6.5.)
c) *Aggregator Parser/Multiplexers.* On transmit, these blocks simply pass frame transmission requests from the Frame Distributor, Marker Generator, and/or Marker Responder to the appropriate Aggregation Port. On receive, these blocks distinguish among Marker Request, Marker Response, and MAC Protocol Data Units (MPDUs), and pass each to the appropriate entity (Marker Responder, Marker Receiver, and Frame Collector, respectively).
d) *Aggregator.* The combination of Frame Distribution and Frame 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 Aggregation 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.

6.2.1 Principles of Link Aggregation

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

a) An Aggregator Client communicates with a set of Aggregation Ports through an Aggregator, which presents a standard ISS interface to the Aggregator Client. The Aggregator binds to one or more Aggregation Ports within a System.
b) It is the responsibility of the Aggregator to distribute frame transmissions from the Aggregator Client to the various Aggregation Ports, and to collect received frames from the Aggregation Ports and pass them to the Aggregator Client transparently.
c) A System may contain multiple Aggregators, serving multiple Aggregator Clients. A given Aggregation Port will bind to (at most) a single Aggregator at any time. An Aggregator Client is served by a single Aggregator at a time.
d) The binding of Aggregation 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 Aggregation Ports within the System to an appropriate Aggregator, and monitoring conditions to determine when a change in aggregation is needed.
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 has to be maintained for certain sequences of frame exchanges between Aggregator Clients (known as *conversations*, see Clause 3). The Frame Distributor ensures that all frames of a given conversation are passed to a single Aggregation Port. For any given Aggregation Port, the Frame Collector is required to pass frames to the Aggregator Client in the order that they are received from that Aggregation Port. The Frame Collector is otherwise free to select frames received from the Aggregation 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 Aggregation Ports within an aggregation, both for load balancing and to maintain availability in the event of link failures. Means are provided (6.5, 6.6) for maintaining frame ordering when such movement takes place.

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

i) Each Aggregation Port is assigned a MAC address, unique over the LAG and the IEEE 802.1Q Bridged LAN (if any) to which the LAG is connected. This MAC address is used as the source address (SA) 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 (DA) for all exchanges, and do not impose any requirement for an Aggregation Port to recognize more than one unicast address on received frames.

j) Each Aggregator is assigned a MAC address, unique over the LAG and the IEEE 802.1Q Bridged LAN (if any) to which the LAG is connected; this address is used as the MAC address of the aggregation from the perspective of the Aggregator Client, both as a SA for transmitted frames and as the destination address (DA) for received frames. The MAC address of the Aggregator may be one of the MAC addresses of an Aggregation Port in the associated LAG (see 6.2.11).

6.2.2 Service interfaces

The Aggregator Client communicates with the Aggregator using the ISS specified in IEEE Std 802.1AC. 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 Aggregation Ports using the same service interface. No new interlayer service interfaces are defined for Link Aggregation.

Since Link Aggregation uses four instances of the ISS, 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 6-2. The prefixes are as follows:

a) *Agg:*, for primitives issued on the interface between the Aggregator 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 Link Aggregation Control (where N is the Port Number associated with the Control Parser/Multiplexer).

d) *DataMuxN:*, for primitives issued on the interface between Control Parser/Multiplexer N and Aggregator Parser/Multiplexer N (where N is the Port Number associated with the Control Parser/Multiplexer).

e) *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).

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

In order to satisfy the needs of higher layers to access physical ports directly, the Protocol Parser/Multiplexer (6.2.7) is provided. In particular, if IEEE Std 802.1AB is implemented in a System supporting Link Aggregation, the LLDP Parser/Multiplexer (6.1.3) will enable to attach zero or more instances of LLDP to each Aggregation Port.

6.2.3 Frame Collector

A Frame Collector is responsible for receiving incoming frames (i.e., AggMuxN:M_UNITDATA.indications) from the set of individual links that form the LAG (through each link's associated Aggregator Parser/Multiplexer) and delivering them to the Aggregator Client. Frames received from a given Aggregation Port are delivered to the Aggregator 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. If a System uses conversation-sensitive frame collection and distribution, then the Frame Collector will discard frames received on Aggregation Ports whose Conversation IDs are not in the list supplied by the Frame Distributor for that Aggregation Port (6.6).

The Frame Collector shall implement the function specified in the state diagram shown in Figure 6-3 and the associated definitions contained in 6.2.3.1.

6.2.3.1 Frame Collector state diagram

6.2.3.1.1 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 Aggregator Client. Value is assigned by management or administration policy.
Value: Integer

6.2.3.1.2 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

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

6.2.3.1.3 Messages

Agg:M_UNITDATA.indication
AggMuxN:M_UNITDATA.indication
: The service primitives used to pass a received frame to a client with the specified parameters.

6.2.3.1.4 State diagram

The architectural models of ISS and Link Aggregation do not make any provision for queuing of frames between the link and the Aggregator Client. However, practical implementations of Link Aggregation will typically incur both queuing 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 Aggregator 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 Aggregator Client or discarded after a CollectorMaxDelay plus the propagation delay of the link. The use of CollectorMaxDelay is further discussed in B.3.

NOTE—Because frame discard due to CollectorMaxDelay is a function of queuing and other entities not specified in this document, the discard action is a requirement of the system, not just of the Frame Collector, and is therefore not shown in Figure 6-3.

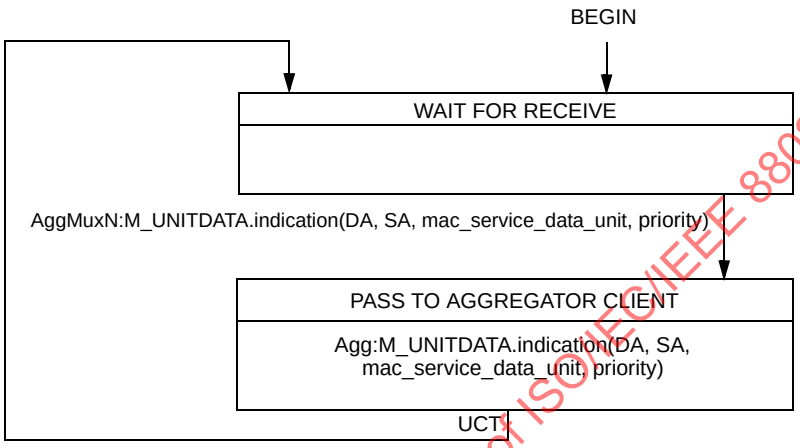


Figure 6-3—Frame Collector state diagram

6.2.4 Frame Distributor

The Frame Distributor is responsible for taking outgoing frames from the Aggregator Client and transmitting them through the set of links that form the LAG. 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 6.2.3, the algorithm shall not cause the following:

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

The preceding 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 Aggregator 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 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 Clause 8 and Annex A.

The Frame Distributor shall implement the function specified in the state diagram shown in Figure 6-4 and the associated definitions contained in 6.2.4.1.

6.2.4.1 Frame Distributor state diagram

6.2.4.1.1 Variables

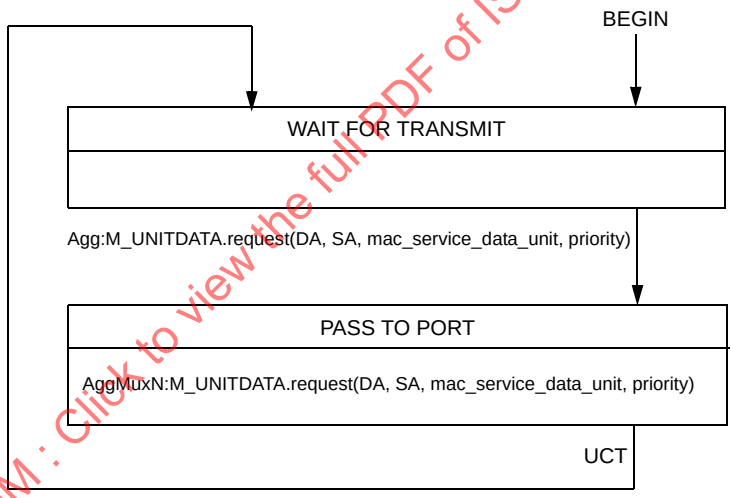
DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.request primitive.

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

6.2.4.1.2 Messages

Agg:M_UNITDATA.request
AggMuxN:M_UNITDATA.request
: The service primitives used to transmit a frame with the specified parameters.

6.2.4.1.3 State diagram



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

Figure 6-4—Frame Distributor state diagram

6.2.5 Marker Generator/Receiver (optional)

The optional Marker Generator is used by the Marker protocol, as specified in 6.5. When implemented and so requested by the distribution algorithm, the Marker Generator shall issue an AggMuxN:M_UNITDATA.request primitive, with a mac_service_data_unit containing a Marker PDU as defined in 6.5.3, to the Aggregation Port associated with the conversation being marked, subject to the timing restrictions for Slow Protocols specified in IEEE Std 802.3-2008, Annex 57A.

The optional Marker Receiver is used by the Marker protocol, as specified in 6.5. It receives Marker Response PDUs from the Aggregator Parser.

6.2.6 Marker Responder

The Marker Responder is used by the Marker protocol, as specified in 6.5. The Marker Responder receives Marker PDUs (generated by a Partner System's Marker Generator), and transmits a Marker Response PDU through the same Aggregation 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 that is not under a Distributed Relay function (DR Function, see 9.2, 9.3) shall implement the Marker Responder as specified in 6.5.4.2, thus ensuring that implementations that need to make use of the protocol can do so.

A Marker Responder that is under a DR Function is not required to respond to a Marker PDU.

NOTE—Since this standard lacks any means for coordinating Marker PDU responses among the Portal Systems comprising a Portal, it is safer to not respond to a Marker PDU, and allow the sender to time out, rather than to return a Marker PDU and run the risk of delivering frames out of order.

6.2.7 Protocol Parser/Multiplexer

A Protocol Parser/Multiplexer is a function of which there are three examples in this standard, as follows:

a) The LLDP Parser/Multiplexer (6.1.3)
b) The Control Parser/Multiplexer (6.2.10)
c) The DRCP Control Parser/Multiplexer (9.4.4)

A Protocol Parser/Multiplexer has three service interfaces, each an instance of the ISS, as follows:

d) One *DownPort* makes use of an instance of the ISS to pass data and control frames to and from lower layers in the protocol stack.
e) One *DataPort* offers an instance of the ISS to a higher level data function.
f) One *ControlPort* offers an instance of the ISS to higher level control functions.

The specification of a given instance of the Protocol Parser/Multiplexer, e.g., the Control Parser/Multiplexer (6.2.10), ties these three notional service interfaces to specific service interfaces, e.g., those in 6.2.2, and provides a definition of the IsControlFrame function ().

On transmission, the Protocol Multiplexer shall provide transparent pass-through of frames from the ControlPort or the DataPort to the DownPort.

On receipt of a frame from its DownPort, the Protocol Parser uses the IsControlFrame function to determine whether the frame is or is not a control frame. It passes control frames up through the ControlPort, and passes all other frames up through the DataPort. The Protocol Parser shall implement the function specified by the state diagram shown in Figure 6-5 and the associated definitions contained in 6.2.7.1.

6.2.7.1 Protocol Parser state diagram

6.2.7.1.1 Functions

IsControlFrame
Given an input frame (DA, SA, mac_service_data_unit, priority), returns TRUE if the frame is a control frame for the ControlPort, or FALSE if it is a data frame for the DataPort.
Value returned: Boolean

6.2.7.1.2 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

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

6.2.7.1.3 Messages

ControlPort:M_UNITDATA.indication
DataPort:M_UNITDATA.indication
DownPort:M_UNITDATA.indication
: The service primitives used to pass a received frame to a client with the specified parameters.

6.2.7.1.4 State diagram

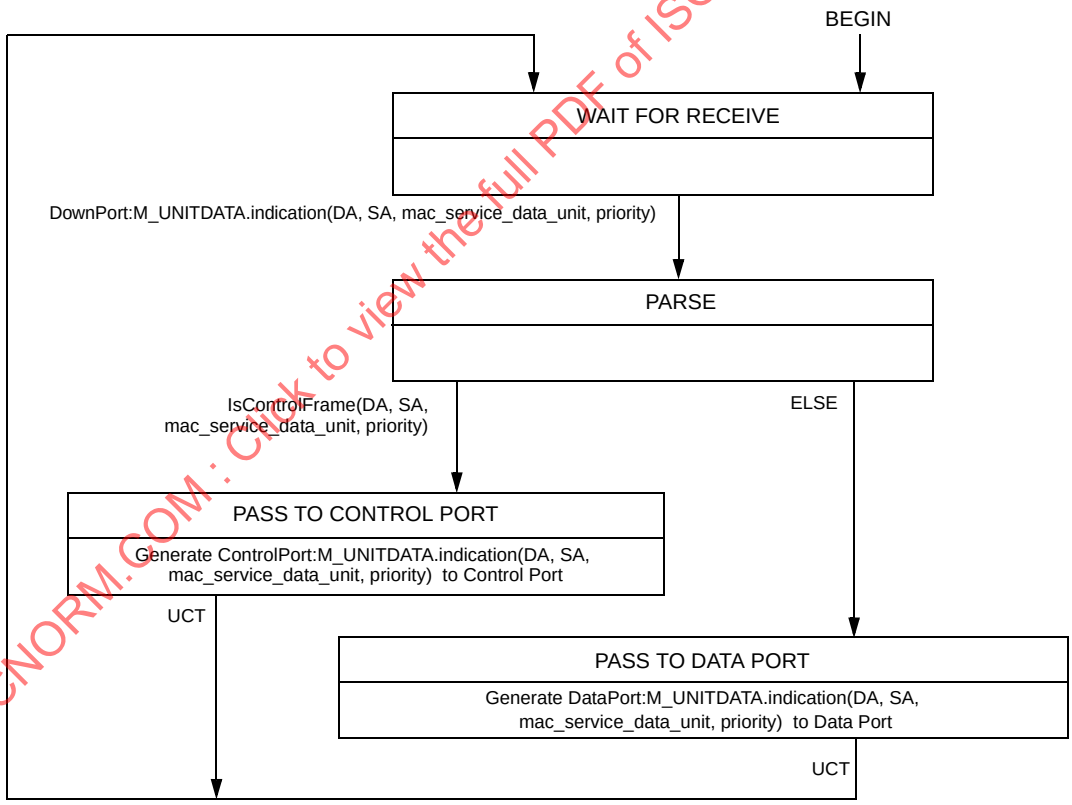


Figure 6-5—Protocol Parser state diagram

6.2.8 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 Aggregation Port specified in the transmission request. The Aggregator Multiplexer shall provide transparent pass-through of frames submitted by the Frame Distributor to the Aggregation Port specified in the transmission request only when the Aggregation Port state is Distributing (see 6.4.15); 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 802.3-2008, Table 57A–2). The Aggregator Parser shall pass all other frames to the Frame Collector for passage to the Aggregator Client only when the Aggregation Port state is Collecting (see 6.4.15); otherwise, such frames shall be discarded. The Aggregator Parser shall implement the function specified in the state diagram shown in Figure 6-6 and the associated definitions contained in 6.2.8.1.

6.2.8.1 Aggregator Parser state diagram

6.2.8.1.1 Constants

Slow_Protocols_Type
: The value of the Slow Protocols Length/Type field. (See IEEE Std 802.3-2008, Annex 57A.)

Marker_subtype
: The value of the Subtype field for the Marker protocol. (See 6.5.3.)
Value: Integer
2

Marker_Information
: The encoding of the Marker Information TLV_type field. (See 6.5.3.)
Value: Integer
1

Marker_Response_Information
: The encoding of the Marker Response Information TLV_type field. (See 6.5.3.)
Value: Integer
2

6.2.8.1.2 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

Protocol_DA
: One of the addresses selected from Table 6-1 determined by the setting of the aAggPortProtocolDA managed object (7.3.2.2.1). A particular instance of link aggregation shall use the same DA for LACP as it does for the Marker protocol.
Value: 48 bits.

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 802.3-2008, 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 6.5.3.)
Value: Integer

BEGIN

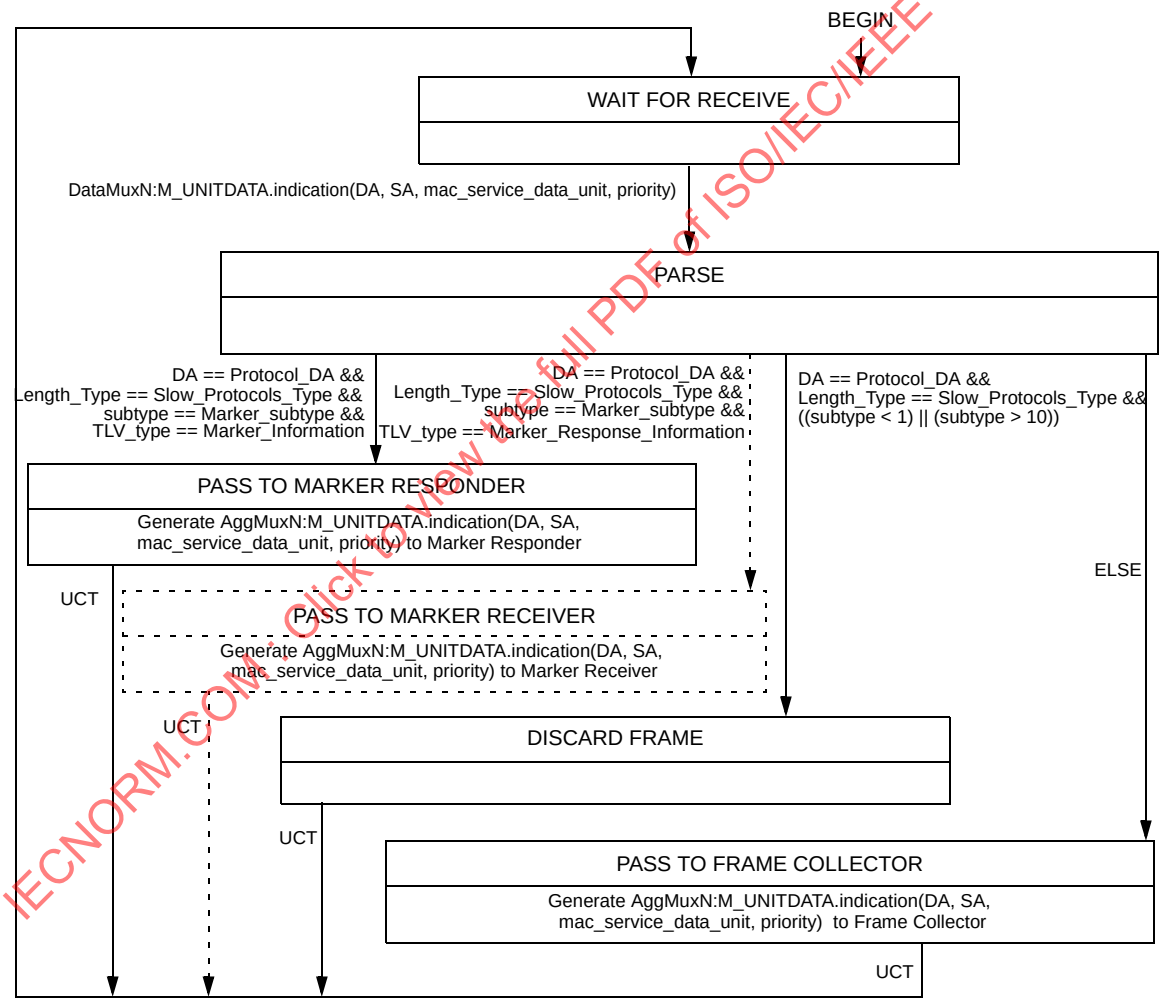
A Boolean variable that is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.
Value: Boolean

6.2.8.1.3 Messages

DataMuxN:M_UNITDATA.indication
AggMuxN:M_UNITDATA.indication

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

6.2.8.1.4 State Diagram



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

Figure 6-6—Aggregator Parser state diagram

6.2.9 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 LAG. A single Aggregator is associated with each LAG. An Aggregator offers an ISS interface to its associated Aggregator Client; access to the physical media by an Aggregator Client is always achieved via an Aggregator. An Aggregator can therefore be considered to be a *logical MAC*, bound to one or more Aggregation Ports, through which the Aggregator Client is provided access to the physical media.

A single, individual MAC address is associated with each Aggregator (see 6.2.11).

An Aggregator is available for use by the Aggregator Client and MAC-Operational at the Aggregator ISS will be True (6.3.12) if the following are all true:

a) It has one or more attached Aggregation Ports.
b) The Aggregator has not been set to a disabled state by administrative action (see 7.3.1.1.13).
c) One or more of the attached Aggregation Ports is Collecting or Distributing (see 7.3.1.1.14).

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 Aggregation Ports in a given System; however, this is not a requirement of this standard. Aggregation of two or more Aggregation Ports consists of changing the bindings between Aggregation Ports and Aggregators such that more than one Aggregation 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 Aggregation Port, and one or more Aggregators that are not bound to any Aggregation Port. An Aggregator that is not bound to any Aggregation Port appears to an Aggregator Client as a MAC interface to an inactive Aggregation Port. During times when the bindings between Aggregation Ports and Aggregators are changing, or as a consequence of particular configuration choices, there may be occasions when one or more Aggregation Ports are not bound to any Aggregator.

6.2.10 Control Parser/Multiplexer

There are N Control Parser/Multiplexers, one for each Aggregation Port. Each Control Parser/Multiplexer is an instance of the Protocol Parser/Multiplexer described in 6.2.7. Specifically:

a) The *DownPort* of the Protocol Parser/Multiplexer is *MacN* (6.2.2) for Control Parser/Multiplexer N.
b) The *ControlPort* of the Protocol Parser/Multiplexer is *CtrlMuxN* (6.2.2) for Control Parser/Multiplexer N.
c) The *DataPort* of the Protocol Parser/Multiplexer is *DataMuxN* (6.2.2) for Control Parser/Multiplexer N.
d) The IsControlFrame function is defined in 6.2.10.1.

6.2.10.1 Control Parser state diagram

6.2.10.1.1 Control Parser Function

IsControlFrame
Returns Boolean value: (DA == Protocol_DA && Length/Type == Slow_Protocols_Type && Subtype == LACP_subtype)

6.2.10.1.2 Constants

Slow_Protocols_Type
The value of the Slow Protocols Length/Type field. (See IEEE Std 802.3-2008, Table 57A–2.)
LACP_subtype
The value of the Subtype field for the Link Aggregation Control Protocol. (See IEEE Std 802.3-2008, Table 57A–3.)
Value: Integer
1

6.2.10.1.3 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

Protocol_DA
: One of the addresses selected from Table 6-1 determined by the setting of the aAggPortProtocolDA managed object (7.3.2.2.1). A particular instance of link aggregation shall use the same DA for LACP as it does for the Marker protocol.
Value: 48 bits.

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 802.3-2008, Annex 57A.)
Value: Integer

6.2.11 Addressing

Each IEEE 802 MAC has an associated individual MAC address, whether that MAC is used for Link Aggregation or not.

Each Aggregator to which one or more Aggregation Ports are attached has an associated individual MAC address (see 6.3.3). The MAC address of the Aggregator may be the individual MAC addresses of one of the MACs in the associated LAG, or it may be a distinct MAC address. The manner in which such addresses are chosen is not otherwise constrained by this standard.

6.2.11.1 Source address (SA)

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 Aggregation Port as the SA in frames transmitted through that Aggregation Port. The Aggregator Client sees only the Aggregator and not the underlying MACs, and therefore uses the Aggregator's MAC address as the SA in transmitted frames. If an Aggregator Client submits a frame to the Aggregator for transmission without specifying a SA, the Aggregator inserts its own MAC address as the SA 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.

6.2.11.2 Destination address

Protocol entities sourcing frames from within the Link Aggregation sublayer (e.g., LACP and the Marker protocol) and the DRCP entities in Clause 9 use one of the MAC addresses listed in Table 6-1 as the DA for such frames. These addresses are in the range of the C-VLAN component reserved addresses (see Table 8-1 in IEEE Std 802.1Q-2014) and the S-VLAN component reserved addresses (see Table 8-2 in IEEE Std 802.1Q-2014). The choice of address used determines the scope of propagation of Link Aggregation control and marker PDUs within a bridged LAN, as follows:

a) The *Nearest Customer Bridge* group address is an address that no conformant C-VLAN component or IEEE 802.1D Bridge forwards. However, this address is relayed by Two-Port Media Access Control (MAC) Relay (TPMR) components and S-VLAN components. Therefore, Link

Aggregation control, Distributed Relay control and marker PDUs received on this address represent information about stations that are not separated from the recipient station by any intervening C-VLAN components or IEEE 802.1D Bridges. There may, however, be TPMR components and/or S-VLAN components in the path between the originating and receiving stations.

NOTE 1—This address makes it possible to communicate information between end stations and/or Customer Bridges and for that communication to be transparent to the presence or absence of TPMRs or S-VLAN components in the communication path. The scope of this address is the same as that of a customer-to-customer MACSec connection.

b) The *Slow_Protocols_Multicast* group address is an address that no conformant TPMR component, S-VLAN component, C-VLAN component, or IEEE 802.1D Bridge can forward. Link Aggregation control, Distributed Relay control, and marker PDUs transmitted using this DA can therefore travel no further than those stations that can be reached via a single individual LAN from the originating station. Link Aggregation control, Distributed Relay control, and marker PDUs received on this address therefore represent information about stations that are attached to the same individual LAN segment as the recipient station.

NOTE 2—This address makes it possible to transmit a Link Aggregation Control, DRCP, or Marker frame containing information specific to a single individual LAN, and for that information not to be propagated further than the extent of that individual LAN.

c) The *Nearest non-TPMR Bridge* group MAC address is an address that no conformant C-VLAN component, S-VLAN component, or IEEE 802.1D Bridge can forward. However, this address is relayed by TPMR components. Therefore, Link Aggregation control, Distributed Relay control and marker PDUs received on this address represent information about stations that are not separated from the recipient station by any intervening C-VLAN component, S-VLAN component, or IEEE 802.1D Bridge. There may, however, be one or more TPMR components in the path between the originating and receiving stations.

NOTE 3—This address makes it possible to communicate information between end stations and/or bridges and for that communication to be transparent to the presence or absence of TPMRs in the transmission path. This address is primarily intended for use within provider bridged networks.

Table 6-1—Link Aggregation protocol destination addresses

Assignment	Value
Nearest Customer Bridge group address	01-80-C2-00-00-00
IEEE 802.3 Slow_Protocols_Multicast group address	01-80-C2-00-00-02
Nearest non-TPMR Bridge group address	01-80-C2-00-00-03

The use and encoding of the Slow_Protocols_Multicast group address are specified in Annex 57A of IEEE Std 802.3-2012.

6.3 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 Aggregation 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 LAG.

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

c) Attaches the Aggregation Port to the Aggregator used by the LAG, and detaches the Aggregation 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 Frame Collector and Frame Distributor.

The operation of Link Aggregation Control involves the following activities, which are described in detail in 6.3.1 through 6.3.16:

e) Checking that candidate links can actually be aggregated.
f) Controlling the addition of a link to a LAG, 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 LAG 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 the following:

i) A globally unique identifier for each System that participates in Link Aggregation (see 6.3.2).
j) A means of identifying the set of capabilities associated with each Aggregation Port and with each Aggregator, as understood by a given System.
k) A means of identifying a LAG 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).

6.3.1 Characteristics of Link Aggregation Control

Link Aggregation Control provides a configuration capability that is the following:

a) **Automatic.** In the absence of manual override controls, an appropriate set of LAGs 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 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 the following:

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.

6.3.2 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 Aggregation Ports of the System. In the case of DRNI (Clause 9), all Portal Systems in a Portal have the same System Identifier, which is provided by the concatenation of the Portal's administrated MAC address (7.4.1.1.4) and the Portal's System Priority (7.4.1.1.5).

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.

6.3.3 Aggregator identification

Each Aggregator to which one or more Aggregation Ports are attached shall be assigned a unique, globally or locally 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 Aggregation Ports. No Aggregator shall be assigned a MAC address that is the same as that of an Aggregation Port bound to a different Aggregator within the System. When receiving frames, an Aggregation Port is never required to recognize more than one unicast address, i.e., the Aggregator's MAC address.

NOTE—This allows 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 Aggregation 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.

6.3.4 Port identification

Link Aggregation Control uses a Port Identifier (Port ID), comprising the concatenation of a Port Priority (7.3.2.1.15) and a Port Number (7.3.2.1.14), to identify the Aggregation Port. Port Numbers (and hence, Port IDs) shall be uniquely assigned within a System. Port Number 0 shall not be assigned to any Aggregation 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.

6.3.5 Capability identification

The ability of one Aggregation 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 the following:

a) The Aggregation 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 Aggregation Port by higher layer protocols (e.g., assignment of Network Layer addresses).
d) Characteristics or limitations of the Aggregation Port implementation itself.

Two Keys shall be associated with each Aggregation 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 an Aggregation 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 Aggregation Port.
f) If the System supports the dynamic manipulation of Keys, as discussed in 6.7.2, either to accurately reflect changes in operational capabilities of the Aggregation 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 Aggregation Ports, it signifies that the set of Aggregation Ports have the potential to aggregate together, subject to the considerations discussed in 6.7.2. When a System assigns an operational Key value to a set of Aggregation Ports, it signifies that, in the absence of other constraints, the current operational state of the set of Aggregation Ports allows any subset of that set of Aggregation Ports (including the entire set) to be aggregated together from the perspective of the System making the assignment. The set of such Aggregation 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 Aggregation 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 Aggregation Port Keys, per item e) and item f) in this subclause. Aggregation 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 Aggregation 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 Aggregation 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 Aggregation 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.

6.3.6 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 (or Portals) and whose Aggregation Ports belong to the same Key Group in each System, or
b) An Individual link.

6.3.6.1 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 Aggregation 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 Aggregation 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 Aggregation 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.

6.3.6.2 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, OUIs or CIDs, successive octets are separated by dashes (-), in accordance with the hexadecimal representation for MAC addresses defined in IEEE Std 802.
d) Parameters of the LAG ID are separated by commas.

For example, consider the parameters for the two Partners in a LAG shown in Table 6-2.

Table 6-2—Example Partner parameters

	Partner SKP	Partner TLQ
System Parameters (S, T)	System Priority = 0x8000 (see 6.4.2.3) System MACaddr= AC-DE-48-03-67-80	System Priority = 0x8000 (see 6.4.2.3) System MACaddr = AC-DE-48-03-FF-FF
Key Parameter (K, L)	Key = 0x0001	Key = 0x00AA
Aggregation Port Parameters (P, Q)	Port Priority = 0x80 (see 6.4.2.3) Port Number = 0x0002	Port Priority = 0x80 (see 6.4.2.3) 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.

6.3.7 Selecting a Link Aggregation Group

Each Aggregation Port is selected for membership in the LAG 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 Aggregation Port's operational parameters.

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

6.3.8 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 LAG. 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.

6.3.9 Attaching a link to an Aggregator

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

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

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 an Aggregator 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 LAG, 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 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 LAG 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.

6.3.10 Signaling readiness to transfer user data

Once a link has been attached to an Aggregator (6.3.9) compatible with the agreed-upon LAG (6.3.8), each Link Aggregation Control entity signals to its peer its readiness to transfer user data to and from the Aggregator's Aggregator 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 LAG. 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 6.5. 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.

6.3.11 Enabling the Frame Collector and Frame Distributor

Initially, both the Frame Collector and Frame Distributor 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. When any Aggregation Port attached to the Frame Collection function is Collecting, the Frame 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 Frame Collector is enabled, the Frame 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 6.4.15).

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

6.3.12 MAC_Operational status

MAC_Operational is the Boolean status in the ISS (11.2 in IEEE Std 802.1AC-2012) that indicates to a higher layer entity that a Service Access Point is (TRUE) or is not (FALSE) available for use. MAC_Operational is an indication from a physical link to Link Aggregation that the link is available and is also an indication from the Link Aggregation Sublayer to the higher layers that the sublayer is available.

MAC_Operational from physical links is an input to the port_enabled variable (6.4.7). Through this variable, a link whose MAC_Operational status is FALSE is removed from a LAG.

The Link Aggregation Sublayer shall present its MAC_Operational status as TRUE to higher layers if and only if its Receive_State and Transmit_State variables both have the value Enabled (6.4.6). The operational state of the Aggregator is "up" (MAC_Operational is TRUE) 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 (7.3.1.1.13) 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" (MAC_Operational is FALSE). The operational state of the Aggregator is reported by the aAggOperState (7.3.1.1.14) managed object.

6.3.13 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.

6.3.14 Detaching a link from an Aggregator

An Aggregation Port may be detached from the Aggregator used by its LAG as a result of protocol (e.g., Key) changes, because of System constraints (e.g., exceeding a maximum allowable number of aggregated links, or device failures) at either end of the link, or because MAC_Operational of an underlying Aggregation Port is FALSE. 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 Aggregation 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 LAG and then attach to a compatible Aggregator, as described in 6.3.7 and 6.3.9.

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 (6.5) or other means (6.6) 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.

6.3.15 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 an Aggregation 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.

6.3.16 Link Aggregation Control state information

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

a) The identifier of the LAG to which it currently belongs.
b) The identifier of the Aggregator associated with that LAG.
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 Frame 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 Collecting and Distributing status of the individual links associated with that Aggregator; i.e., if one or more links in the LAG 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 Aggregator Client, or conversely, the point at which it ceases to be available.

6.4 Link Aggregation Control Protocol

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 LAG to which the link belongs, move the link to that LAG, and enable its transmission and reception functions in an orderly manner.

6.4.1 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 Aggregation Port, that can take the value Active LACP or Passive LACP. Passive LACP indicates the Aggregation 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 Aggregation 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.

6.4.2 LACPDU structure and encoding

6.4.2.1 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.

6.4.2.2 Encapsulation of LACPDUs in frames

An LACPDU is encoded in the mac_service_data_unit parameter of an M_UNITDATA.request or M_UNITDATA.indication. The first octets of the mac_service_data_unit are a protocol identifier, followed by the LACPDU, followed by padding octets, if any, as required by the underlying MAC service.

Where the ISS instance used to transmit and receive frames is provided by a media access control method that can support EtherType encoding directly (e.g., is an IEEE 802.3 MAC), the protocol identifier is two octets in length, and the value is the Slow Protocols EtherType (hexadecimal 88-09).

Where the ISS instance is provided by a media access method that cannot directly support EtherType encoding (e.g., is an IEEE 802.11 MAC), the EtherType protocol identifier is encoded according to the rule for a Subnetwork Access Protocol (Clause 9 of IEEE Std 802-2014) that encapsulates Ethernet frames over logical link control (LLC), and comprises the SNAP header (hexadecimal AA-AA-03) followed by the SNAP PID (hexadecimal 00-00-00) followed by the Slow Protocols EtherType (hexadecimal 88-09).

6.4.2.3 LACPDU structure

The LACPDU structure shall be as shown in Figure 6-7 and as further described in the following field definitions:

a) *Subtype*. The Subtype field identifies the specific Slow Protocol being encapsulated. LACPDUs carry the Subtype value 0x01.
b) *Version Number.* This identifies the LACP version; Version 1 implementations conformant to this standard carry the value 0x01 and Version 2 implementations conformant to this standard carry the value of 0x02.
c) *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.
d) *Actor_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Actor information uses a length value of 20 (0x14).
e) *Actor_System_Priority.* The priority assigned to this System (by management or administration policy), encoded as an unsigned integer.
f) *Actor_System.* The MAC address component of Actor's System ID.
g) *Actor_Key.* The operational Key value assigned to the Aggregation Port by the Actor, encoded as an unsigned integer.
h) *Actor_Port_Priority.* The priority assigned to this Aggregation Port by the Actor (the System sending the PDU; assigned by management or administration policy), encoded as an unsigned integer.
i) *Actor_Port.* The Port Number assigned to the Aggregation Port by the Actor (the System sending the PDU), encoded as an unsigned integer.

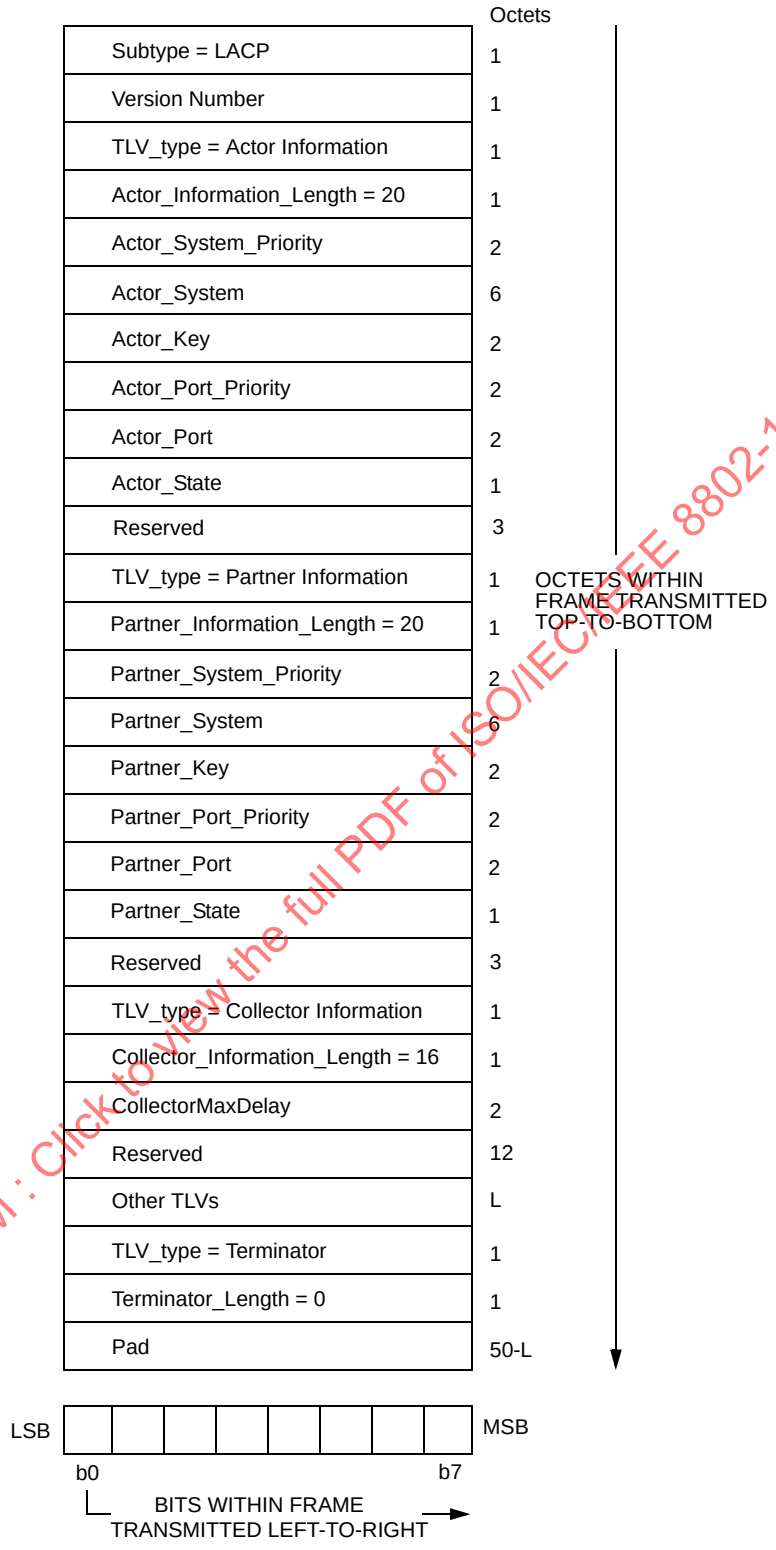
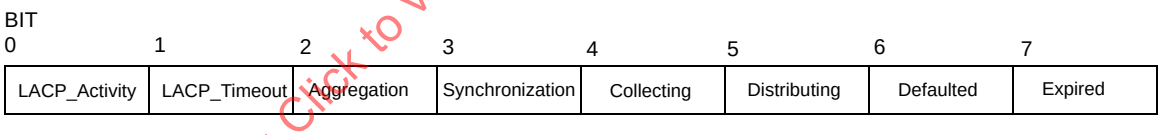


Figure 6-7—LACPDU structure

j) *Actor_State.* The Actor's state variables for the Aggregation Port, encoded as individual bits within a single octet, as follows and as illustrated in Figure 6-8:
   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 LAG, the group has been associated with a compatible Aggregator, and the identity of the LAG 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 LAG.
   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 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 1—The received values of Defaulted and Expired state are not used by LACP; however, knowing their values can be useful when diagnosing protocol problems.



NOTE 2—Bit ordering within this field is as specified in 6.4.2.1.

Figure 6-8—Bit encoding of the Actor_State and Partner_State fields

k) *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.
l) *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.
m) *Partner_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Partner information uses a length value of 20 (0x14).
n) *Partner_System_Priority.* The priority assigned to the Partner System (by management or administration policy), encoded as an unsigned integer.
o) *Partner_System.* The MAC address component of the Partner's System ID.
p) *Partner_Key.* The operational Key value assigned to the Aggregation Port associated with this link by the Partner, encoded as an unsigned integer.

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

r) *Partner_Port.* The Port Number associated with this link assigned to the Aggregation Port by the Partner, encoded as an unsigned integer.

s) *Partner_State.* The Actor's view of the Partner's state variables, depicted in Figure 6-8 and encoded as individual bits within a single octet, as defined for Actor_State.

t) *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.

u) *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.

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

w) *CollectorMaxDelay.* This field contains the value of CollectorMaxDelay (6.2.3.1.1) 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).

x) *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.

y) *Other TLVs.* This field contains additional TLVs (Version 2 TLVs are listed in 6.4.2.4). The length *L* equals the sum of the lengths of all individual additional TLVs. In Version 1 implementations of this standard, *L* shall be 50 octets or less to comply with the fixed frame size of 128 octets. Version 2 and later versions of the protocol do not impose such constraint and *L* can have a maximum length that is only limited by the supporting MAC's maximum service data unit size (for 802.3 media, *L* can be up to 1438 octets long). Version 2 or higher LACPDUs containing additional TLVs of a total length *L* larger than 50 octets are Long LACPDUs.

z) *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.

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

NOTE 3—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.

ab) *Pad*. These 50-*L* octets are required in order to comply with the 128 octets limit imposed by Version 1 implementations. They are ignored on receipt and are transmitted as zeros to claim compliance with Version 1 of this protocol. This field is not used in Long LACPDUs.

NOTE 4—Previous versions of this standard (IEEE Std 802.1AX-2008 and earlier) forced a fixed frame size of 128 octets on IEEE 802.3 media, regardless of the version of the protocol. This version of the standard does not impose such constraint on the maximum frame size as processing limitations that were associated with earlier constraints are not valid any more and the protocol is in general applicable to any media that supports the ISS. In any case, 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 without having to force a fixed frame size. 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.

6.4.2.4 Version 2 TLVs

Table 6-3 provides a list of all TLVs that are applicable for Version 2 LACP. They support the Conversation-sensitive frame collection and distribution functionality described in 6.6.

Table 6-3—Type field values of Version 2 TLVs

TLV	Type field
Port Algorithm TLV	0x04
Port Conversation ID Digest TLV	0x05
Port Conversation Mask-1	0x06
Port Conversation Mask-2	0x07
Port Conversation Mask-3	0x08
Port Conversation Mask-4	0x09
Port Conversation Service Mapping TLV	0x0A

6.4.2.4.1 Port Algorithm TLV

This TLV is required to support the Conversation-sensitive LACP operation (6.6.2) and shall be present on all Version 2 LACPDU exchanges. The Port Algorithm TLV structure shall be as shown in Figure 6-9 and as further described in the following field definitions:

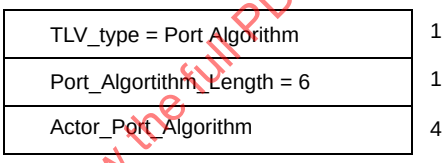


Figure 6-9—Port Algorithm TLV

a) *TLV_type = Port Algorithm.* This field indicates the nature of the information carried in this TLV-tuple. The Port Algorithm TLV is identified by the integer value 0x04.
b) *Port_Algorithm_Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Algorithm TLV uses a length value of 6 (0x06).
c) *Actor_Port_Algorithm.* This field contains the value of the algorithm used to assign frames to Port Conversation IDs. It consists of the 3-octet organizationally unique identifier (OUI) or Company Identifier (CID) identifying the organization that is responsible for this algorithm and one following octet used to identify the Port Algorithm by that organization. It is always set equal to aAggPortAlgorithm (7.3.1.1.33). Table 6-4 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings.

Table 6-4—IEEE Port Algorithms

Port Algorithm field	Value
Unspecified distribution algorithm	0
Distribution based on C-VIDs	1
Distribution based on S-VIDs	2
Distribution based on I-SIDs	3
Distribution based on TE-SIDs	4
Distribution based on ECMP Flow Hash	5
Reserved	6–255

6.4.2.4.2 Port Conversation ID Digest TLV

This TLV is required to support the Conversation-sensitive LACP operation (6.6.2) and shall be present on all Version 2 LACPDU exchanges. The Port Conversation ID Digest TLV structure shall be as shown in Figure 6-10 and as further described in the following field definitions:

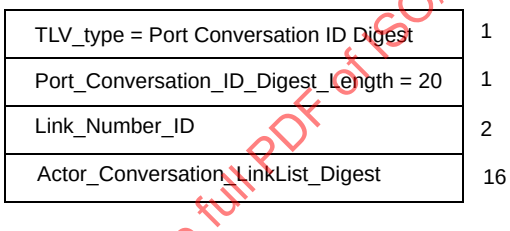


Figure 6-10—Port Conversation ID Digest TLV

a) *TLV_type = Port Conversation ID Digest.* This field indicates the nature of the information carried in this TLV-tuple. The Port Conversation ID Digest TLV is identified by the integer value 0x05.
b) *Port_Conversation_ID_Digest_Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation ID Digest TLV uses a length value of 2 (0x14).
c) *Link_Number_ID.* This field contains the operational value of the Link_Number_ID that is assigned to this Aggregation Port in order to identify Link_Number_ID configuration errors.
d) *Actor_Conversation_LinkList_Digest.* This field contains the value of the MD5 digest Actor_Conversation_LinkList_Digest () for exchange with the Partner System.

6.4.2.4.3 Port Conversation Mask TLVs

There are four Port Conversation Mask TLVs, as follows:

a) Port Conversation Mask-1 TLV
b) Port Conversation Mask-2 TLV
c) Port Conversation Mask-3 TLV
d) Port Conversation Mask-4 TLV

If any of the Port Conversation Mask TLVs are to be carried in an LACPDU then all four shall be carried together and placed in the same order as the preceding list. These TLVs are required to support the Conversation-sensitive LACP operation (6.6.2), and they shall be present on all Long LACPDU exchanges.

The Port Conversation Mask-1 TLV structure shall be as shown in Figure 6-11 and as further described in the following field definitions:

TLV_type = Port Conversation Mask-1	1
Port_Conversation_Mask_1_Length = 131	1
Port_Conversation_Mask_State	1
Port_Oper_Conversation_Mask_1	128

Figure 6-11—Port Conversation Mask-1 TLV

e) *TLV_type = Port Conversation Mask-1*. This field indicates the nature of the information carried in this TLV-tuple. The Port Conversation Mask-1 TLV is identified by the integer value 0x06.

f) *Port_Conversation_Mask_1_Length*. This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Mask-1 TLV uses a length value of 131 (0x83).

g) *Port_Conversation_Mask_State*. The Port Conversation Mask state variables for the Aggregation Port, encoded as individual bits within a single octet, as follows and as illustrated in Figure 6-12:

1) *ActPar_Sync* is encoded in bit 0. This flag indicates if the Port Conversation Mask used by the Actor's Frame Distributor is the same or not as that used by the Partner's Frame Distributor. TRUE (encoded as a 1) if Partner_Oper_Conversation_Mask == Port_Oper_Conversation_Mask. Its value is otherwise FALSE (encoded as a 0);

2) *Portal System Isolated (PSI*) is encoded in bit 1. This flag is only applicable for Portal Systems (Clause 9) and is used to indicate if the Portal System is isolated from the other Portal Systems within the Portal (). TRUE (encoded as a 1) if DRF_Neighbor_Oper_DRCP_State.IPP_Activity == FALSE on all IPPs on this Portal System. Its value is otherwise FALSE (encoded as a 0);

3) *Discard Wrong Conversation (DWC*) is encoded in bit 2. This flag is used to indicate if the Aggregator will discard frames with incorrect Port Conversation IDs. It is encoded as a 1 if Discard_Wrong_Conversation == TRUE and as 0 otherwise;

4) All other bits in the octet are reserved for future use. They are transmitted as zero and are ignored on receipt.

BIT 0	1	2	3	4	5	6	7
ActPar_Sync	PSI	DWC	Reserved	Reserved	Reserved	Reserved	Reserved

NOTE—Bit ordering within this field is as specified in 6.4.2.1.

Figure 6-12—Bit encoding of the Conversation_Mask_State fields

h) *Port_Oper_Conversation_Mask_1*. This field contains the Boolean values of the mask for the first 1024 indexed Port Conversation IDs of the Port_Oper_Conversation_Mask Boolean vector encoded in increasing Port Conversation ID order starting from Port Conversation ID 0 up to Port Conversation ID 1023.

The Port Conversation Mask-2 TLV structure shall be as shown in Figure 6-13 and as further described in the following field definitions:

TLV_type = Port Conversation Mask-2	1
Port_Conversation_Mask_2_Length = 130	1
Port_Oper_Conversation_Mask_2	128

Figure 6-13—Port Conversation Mask-2 TLV

i) *TLV_type = Port Conversation Mask-2.* This field indicates the nature of the information carried in this TLV-tuple. The Port Conversation Mask-2 TLV is identified by the integer value 0x07.
j) *Port_Conversation_Mask_2_Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Mask-2 TLV uses a length value of 130 (0x82).
k) *Port_Oper_Conversation_Mask_2.* This field contains the Boolean values of the mask for the second 1024 indexed Port Conversation IDs of the Port_Oper_Conversation_Mask Boolean vector encoded in increasing Port Conversation ID order starting from Port Conversation ID 1024 up to Port Conversation ID 2047.

The Port Conversation Mask-3 TLV structure shall be as shown in Figure 6-14 and as further described in the following field definitions:

TLV_type = Port Conversation Mask-3	1
Port_Conversation_Mask_3_Length = 130	1
Port_Oper_Conversation_Mask_3	128

Figure 6-14—Port Conversation Mask-3 TLV

l) *TLV_type = Port Conversation Mask-3.* This field indicates the nature of the information carried in this TLV-tuple. The Port Conversation Mask-3 TLV is identified by the integer value 0x08.
m) *Port_Conversation_Mask_3_Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Mask-3 TLV uses a length value of 130 (0x82).
n) *Port_Oper_Conversation_Mask_3.* This field contains the Boolean values of the mask for the third 1024 indexed Port Conversation IDs of the Port_Oper_Conversation_Mask Boolean vector encoded in increasing Port Conversation ID order starting from Port Conversation ID 2048 up to Port Conversation ID 3071.

The Port Conversation Mask-4 TLV structure shall be as shown in Figure 6-15 and as further described in the following field definitions:

TLV_type = Port Conversation Mask-4	1
Port_Conversation_Mask_4_Length = 130	1
Port_Oper_Conversation_Mask_4	128

Figure 6-15—Port Conversation Mask-4 TLV

o) *TLV_type = Port Conversation Mask-4.* This field indicates the nature of the information carried in this TLV-tuple. The Port Conversation Mask-4 TLV is identified by the integer value 0x09.
p) *Port_Conversation_Mask_4_Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Mask-4 TLV uses a length value of 130 (0x82).

q) *Port_Oper_Conversation_Mask_4*. This field contains the Boolean values of the mask for the final 1024 indexed Port Conversation IDs of the Port_Oper_Conversation_Mask Boolean vector encoded in increasing Port Conversation ID order starting from Port Conversation ID 3072 up to Port Conversation ID 4095.

6.4.2.4.4 Port Conversation Service Mapping TLV

This TLV is only required when the Service IDs are different from the Conversation ID. The Port Conversation Service Mapping TLV structure shall be as shown in Figure 6-16 and as further described in the following field definitions:

TLV_type = Port Conversation Service Mapping Digest	1
Port_Conversation_Service_Mapping_Digest_Length = 18	1
Actor_Conversation_Service_Mapping_Digest	16

Figure 6-16—Port Conversation Service Mapping TLV

a) *TLV_type = Port Conversation Service Mapping Digest.* This field indicates the nature of the information carried in this TLV-tuple. The Port Conversation Service Mapping TLV is identified by the integer value 0x0A.
b) *Port_Conversation_Service Mapping_Digest_Length.* This field indicates the length (in octets) of this TLV-tuple. The Port Conversation Service Mapping TLV uses a length value of 18 (0x12).
c) *Actor_Conversation_Service_Mapping_Digest.* This field contains the value of the MD5 digest computed from aAggAdminServiceConversationMap[] (7.3.1.1.38) for exchange with the Partner System.

6.4.3 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-Aggregation Port basis; any deviations from per-Aggregation 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 6.4.7), generated by the state machines as necessary.

The state machines are as follows:

a) *Receive machine (RX*—6.4.12*).* 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 Aggregation Port can now be safely used, either in an aggregation with other Aggregation Ports or as an individual Aggregation 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* (6.4.13). 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* (6.4.14). The Selection Logic is responsible for selecting the Aggregator to be associated with this Aggregation Port.

d) *Mux machine (MUX*—6.4.15*)*. This state machine is responsible for attaching the Aggregation Port to a selected Aggregator, detaching the Aggregation Port from a deselected Aggregator, and for turning collecting and distributing at the Aggregation Port on or off as required by the current protocol information.

e) *Transmit machine (TX*—6.4.16*)*. This state machine handles the transmission of LACPDUs, both on demand from the other state machines, and on a periodic basis.

Figure 6-17 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 Aggregation Port or Ports can affect the operation of an Aggregation Port's state machine. See the definition of the Ready and Ready_N variables in 6.4.8.

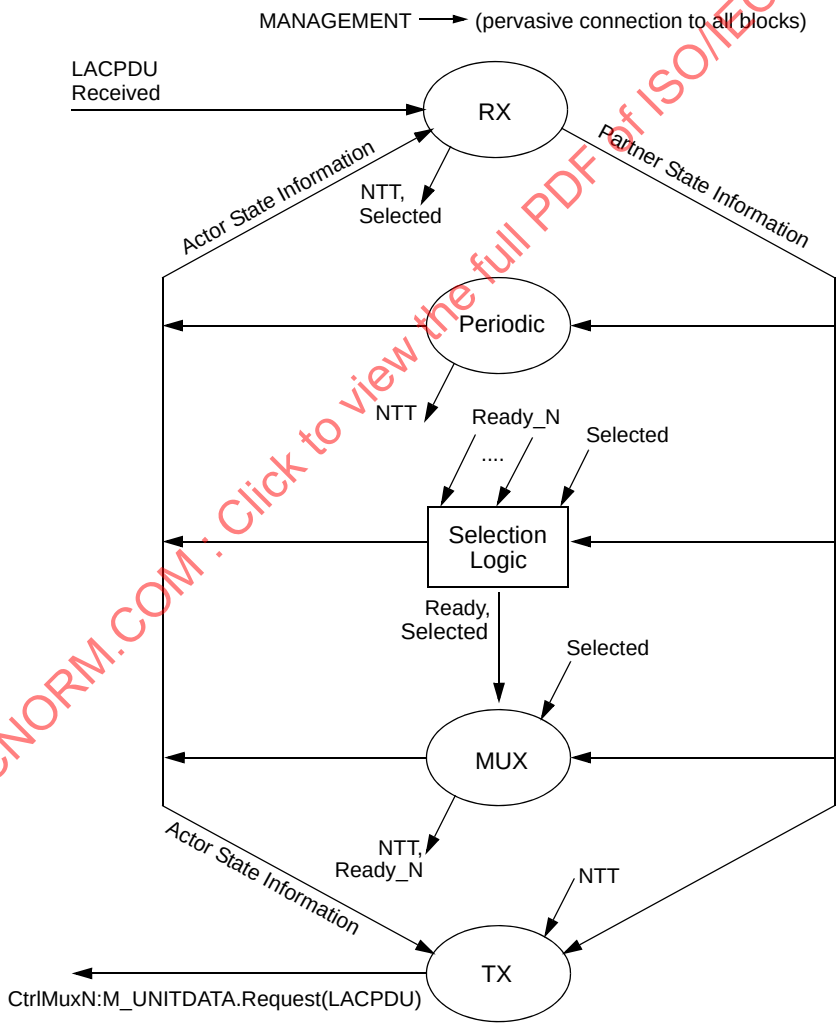


Figure 6-17—Interrelationships among state machines

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

f) *Actor and Partner Churn Detection machines* (6.4.17). 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 6.7), these state machines simply detect the presence of such a condition and signal its existence to management.

6.4.4 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

Actor_System_LACP_Version
: The Version number of the Actor's LACP implementation.
Value: Integer

6.4.5 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.

6.4.6 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 Aggregation Port attached to this Aggregator is not capable of aggregation with any other Aggregation Port.
FALSE if the Aggregation Port(s) attached to this Aggregator are capable of aggregation with other Aggregation 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 Aggregation 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 Aggregation 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 Aggregation Ports, this variable is set to zero.
Value: Integer

Receive_State
The Receive_State of the Aggregator will be Enabled if one or more Aggregation Ports attached to the Aggregator are Collecting (i.e., Actor_Oper_Port_State.Collecting is TRUE for any Aggregation 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 Aggregation Ports attached to the Aggregator are Distributing (i.e., Actor_Oper_Port_State.Distributing is TRUE for any Aggregation Port). Otherwise, Transmit_State is Disabled.
Values: Enabled or Disabled

LAG_Ports
The set of Aggregation Ports that belong to the Link Aggregation Group.
Value: Integer Array

6.4.7 Variables associated with each Aggregation Port

Actor_Port_Number
The Port Number assigned to the Aggregation Port.
Value: Integer
Assigned by administrator or System policy.

Actor_Port_Priority
The Port Priority value assigned to the Aggregation 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 Aggregation 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 Aggregation Port by administrator or System policy.
Value: Integer

Actor_Oper_Port_Key
The operational value of Key assigned to this Aggregation 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 6.4.2.3:
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 6.4.2.3:
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 6.4.2.3:
    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 6.4.2.3:

LACP_Activity
LACP_Timeout
Aggregation
Synchronization
Collecting
Distributing
Defaulted
Expired

Value: 8 bits

port_enabled

A variable indicating that the link has been established and the Aggregation Port is operable.
Value: Boolean

TRUE if the Aggregation Port is operable (MAC_Operational == TRUE).
FALSE otherwise.

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

Partner_LACPDU_Version_Number

The Version number of the LACP as reported by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the default value 1. This variable is only applicable to Version 2 or higher implementations of the protocol.
Value: Integer

enable_long_pdu_xmit

A variable indicating that Long LACPDUs can be transmitted. TRUE when Long LACPDUs can be transmitted, FALSE when only fixed sized LACPDUs (110 octets) can be transmitted. The variable is only applicable to Version 2 or higher implementations of the protocol.
Value: Boolean

6.4.8 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 Aggregation Port is operating the LACP. If the link is not a point-to-point link, the value of LACP_Enabled shall be FALSE. Otherwise, the value of LACP_Enabled shall be TRUE.
Value: Boolean

NOTE 1—Among IEEE 802.3 links, full-duplex links are always point-to-point links. Previous versions of Link Aggregation (IEEE Std 802.1AX-2008, IEEE Std 802.1AXbk™-2012, and IEEE Std 802.3-2005, Clause 43), which were restricted only to IEEE 802.3 links, specified that the state of LACP_Enabled depended upon whether the link was operating in full-duplex or half-duplex mode. It is the point-to-point characteristic of the link that is essential for the operation of Link Aggregation, hence the preceding definition of LACP_Enabled.

actor_churn

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

Value: Boolean

Ready_N

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

Value: Boolean

Ready

The Selection Logic asserts Ready TRUE when the values of Ready_N for all Aggregation Ports that are waiting to attach to a given Aggregator are TRUE. If any of the values of Ready_N for the Aggregation Ports that are waiting to attach to that Aggregator are FALSE, or if there are no Aggregation 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 Aggregation Port from being enabled as part of the aggregation, and so the Aggregation Port is being held in a standby condition. This variable can only be set to SELECTED or STANDBY by the operation of the Aggregation Port's Selection Logic. It can be set to UNSELECTED by the operation of the Aggregation Port's Receive machine, or by the operation of the Selection Logic associated with another Aggregation Port.

NOTE 2—Setting Selected UNSELECTED in the Selection Logic associated with another Aggregation Port occurs if the Selection Logic determines that the other Aggregation Port has a stronger claim to attach to this Aggregation Port's current Aggregator.

Value: SELECTED, UNSELECTED, or STANDBY

port_moved

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

Value: Boolean

6.4.9 Functions

recordPDU

This function records the parameter values for the Actor carried in a received LACPDU (Actor_Port_Number, 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 and Partner_Oper_Port_State.Synchronization 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 Aggregation 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 Aggregation 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 Aggregation Port's Control Parser/Multiplexer to be detached from the Aggregator Parser/Multiplexer of the Aggregator to which the Aggregation Port is currently attached.

Enable_Collecting

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

Disable_Collecting

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

Enable_Distributing

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

Disable_Distributing

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

Enable_Collecting_Distributing

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

Disable_Collecting_Distributing

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

recordVersionNumber

This function records the Partner's LACP implementation version as carried in the received LACPDU's *Version Number* field and updates the value of the variable Partner_LACPDU_Version_Number with the received value. This function is only applicable to Version 2 or higher implementations of this protocol.

6.4.10 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 6.4.4).

actor_churn_timer
: This timer is used to detect Actor churn states. It is started using the value Churn_Detection_Time (see 6.4.4).

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 6.4.4), 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 6.4.4).

wait_while_timer
: This timer provides hysteresis before performing an aggregation change, to allow all links that will join this LAG to do so. It is started using the value Aggregate_Wait_Time (see 6.4.4).

6.4.11 Messages

CtrlMuxN:M_UNITDATA.indication(LACPDU)
: This message is generated by the Control Parser as a result of the reception of a LACPDU, formatted as defined in 6.4.2.

6.4.12 Receive machine

The Receive machine shall implement the function specified in Figure 6-18 with its associated parameters (6.4.4 through 6.4.11).

On receipt of a LACPDU, the state machine enters the CURRENT state. The update_Selected function sets the Selected variable to UNSELECTED if the Actor's view of the Partner's operational parameters is not up to date. The Selected variable is used by the Mux machine (6.4.15).

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 6.4.2.3. Version 2 or higher implementations record as well the Version Number of the received LACPs in order to check if Long LACPDUs can be transmitted on this link (6.4.18).

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

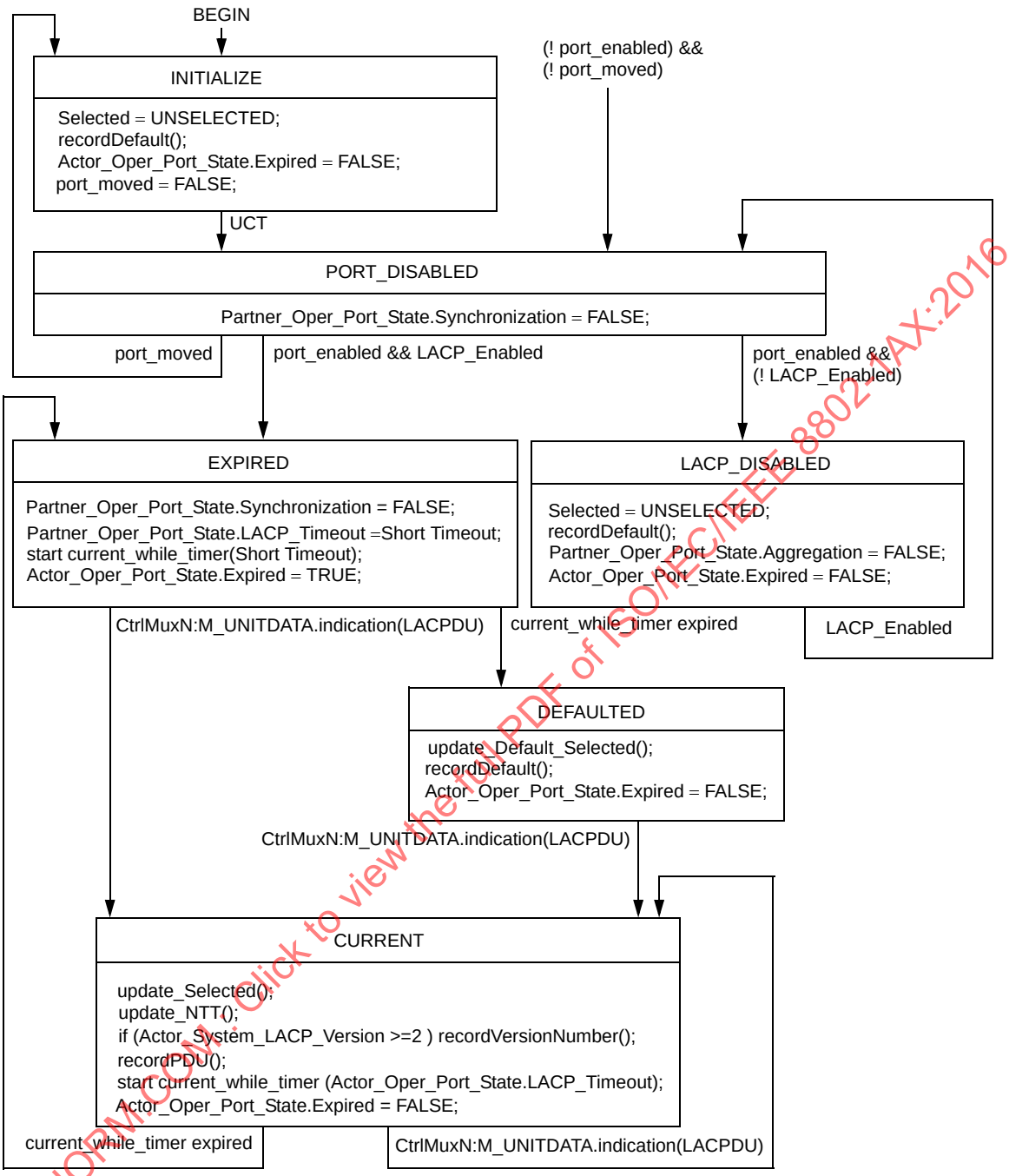


Figure 6-18—Receive machine state diagram

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 reestablish 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 update_Default_Selected function sets the Selected variable to UNSELECTED if

the LAG has changed. 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. Since all operational parameters are now set to locally administered values, there can be no disagreement as to the LAG, so the Partner_Oper_Port_State.Synchronization variable is set to TRUE.

If the Aggregation Port becomes inoperable and the BEGIN variable is not asserted, 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 Aggregation Port is still connected to the same Partner and Partner Aggregation Port when it becomes operable again, there will be no disturbance caused to higher layers by unnecessary re-configuration. If the same Actor System ID and Aggregation Port are seen in a LACPDU received on a different Aggregation 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 Aggregation 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 Aggregation Port is attached to a link that is not a point-to-point link, the operation of LACP is disabled on the Aggregation 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 Aggregation Port is forced to operate as an Individual Aggregation Port, as the value of Partner_Oper_Port_State.Aggregation is forced to Individual. Exit from this state occurs on a BEGIN, Port Disabled, or LACP_Enabled event.

6.4.13 Periodic Transmission machine

The Periodic Transmission machine shall implement the function specified in Figure 6-19 with its associated parameters (6.4.4 through 6.4.11).

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.

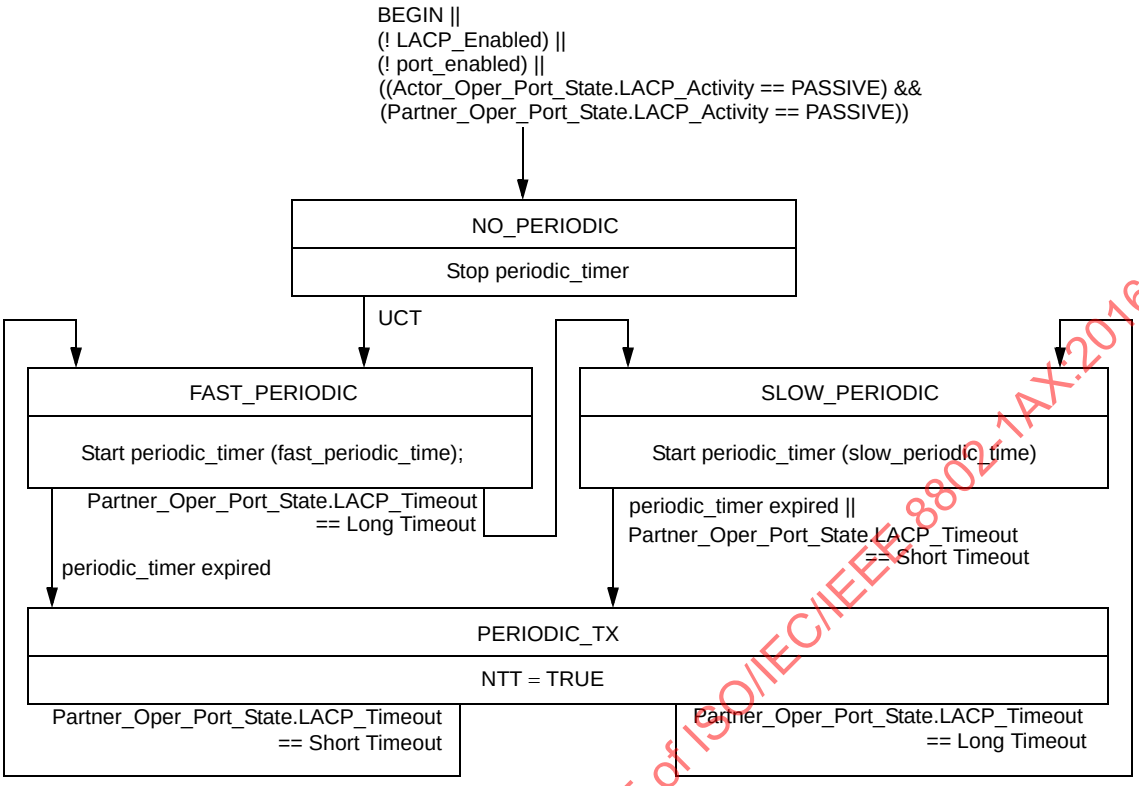


Figure 6-19—Periodic Transmission machine state diagram

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 Aggregation Port or that the Aggregation 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.

6.4.14 Selection Logic

The Selection Logic selects a compatible Aggregator for an Aggregation Port, using the Aggregation 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 Aggregation 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 Aggregation Ports to be moved to the correct Aggregators during reconfiguration.

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

NOTE 2—An Aggregation 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 Aggregation Port is detached from any prior Aggregator; other Aggregators may become free while the Aggregation Port is detaching, and other Aggregation 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 6.4.14.1.
b) The recommended default operation of the Selection Logic is described in 6.4.14.2.

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 6.4.14.1.

6.4.14.1 Selection Logic—Requirements

Aggregation is represented by an Aggregation 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 Aggregation Port shall be assigned an operational Key (6.3.5). Aggregation Ports that can aggregate together are assigned the same operational Key as the other Aggregation Ports with which they can aggregate; Aggregation Ports that cannot aggregate with any other Aggregation 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) An Aggregation 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), Aggregation Ports that are members of the same LAG (i.e., two or more Aggregation 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 Aggregation Ports that are members of the same LAG, but are connected together by the same link, shall not select the same Aggregator (i.e., if a loopback condition exists between two Aggregation Ports, they shall not be aggregated together. For both Aggregation Ports, the Actor System ID is the same as the Partner System ID; also, for Aggregation Port A, the Partner's Port Identifier is Aggregation Port B, and for Aggregation Port B, the Partner's Port Identifier is Aggregation Port A).

   NOTE 1—This exception condition prevents the formation of an aggregated link, comprising two ends of the same 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 Aggregation Port A is looped back to Aggregation Port C and Aggregation Port B is looped back to Aggregation 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 Aggregation Port that is required to be Individual (i.e., the operational state for the Actor or the Partner indicates that the Aggregation Port is Individual) shall not select the same Aggregator as any other Aggregation Port.
i) Any Aggregation Port that is Aggregateable shall not select an Aggregator to which an Individual Aggregation Port is already attached.
j) If the preceding conditions result in a given Aggregation Port being unable to select an Aggregator, then that Aggregation Port shall not be attached to any Aggregator.

k) If there are further constraints on the attachment of Aggregation Ports that have selected an Aggregator, those Aggregation Ports may be selected as standby in accordance with the rules specified in 6.7.1. Selection or deselection of that Aggregator can cause the Selection Logic to reevaluate the Aggregation 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 Aggregation Port, determined from these operational parameters, to locate the correct Aggregator to which to attach the Aggregation Port.

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

NOTE 2—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 reallocate Aggregation Ports to other Aggregators.

n) Once the correct Aggregator has been determined, the variable Selected shall be set to SELECTED or to STANDBY (6.4.8, 6.7.1).

NOTE 3—If Selected is SELECTED, the Mux machine will start the process of attaching the Aggregation Port to the selected Aggregator. If Selected is STANDBY, the Mux machine holds the Aggregation 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 Aggregation Ports that are waiting to attach to the same Aggregator (see 6.4.8).

p) Where the selection of a new Aggregator by an Aggregation Port, as a result of changes to the selection parameters, results in other Aggregation Ports in the System being required to reselect their Aggregators in turn, this is achieved by setting Selected to UNSELECTED for those other Aggregation Ports that are required to reselect their Aggregators.

NOTE 4—The value of Selected is set to UNSELECTED by the Receive machine for the Aggregation Port when a change of LAG ID is detected.

q) An Aggregation Port shall not be enabled for use by the Aggregator Client until it has both selected and attached to an Aggregator.

r) An Aggregation Port shall not select an Aggregator, which has been assigned to a Portal (Clause 9), unless the Partner_Oper_Key of the associated LAG ID is equal to the lowest numerical value of the set comprising the values of the DRF_Home_Oper_Partner_Aggregator_Key, the DRF_Neighbor_Oper_Partner_Aggregator_Key, and the DRF_Other_Neighbor_Oper_Partner_Aggregator_Key, on each IPP on the Portal System, where any variable having its most significant bits set to 00 is excluded (corresponding to accepting only variables that have an integer value that is larger than 16383).

s) An Aggregation Port shall not select an Aggregator, which has been assigned to a Portal (Clause 9), if its Portal's System Identifier is set to a value that is numerically lower than the Partner's System Identifier, PSI == TRUE, the most significant two bits of Partner_Oper_Key are equal to the value 2 or 3 and the two least significant bits of the Aggregation Port's Partner_Oper_Port_Priority are equal to the value 2 or 3. This is to prevent network partition due to isolation of the Portal Systems in the interconnected Portals (Clause 9).

6.4.14.2 Selection Logic—Recommended default operation

The recommended default behavior provides an element of determinism (i.e., history independence) in the assignment of Aggregation 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 6.4.14.1. In particular, it is possible within these requirements to support implementations that provide fewer Aggregators than Aggregation Ports, as well as implementations designed to minimize configuration changes at the expense of less deterministic behavior.

Each Aggregation Port has an Aggregator associated with it (i.e., the number of Aggregators in the System equals the number of Aggregation Ports supported). Each Aggregation Port/Aggregator pair is assigned the same operational Key and Port Number. When there are multiple Aggregation Ports in an aggregation, the Aggregator that the set of Aggregation Ports selects is the Aggregator with the same Port Number as the lowest numbered Aggregation Port in the aggregation. Note that this lowest numbered Aggregation 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 6-20.

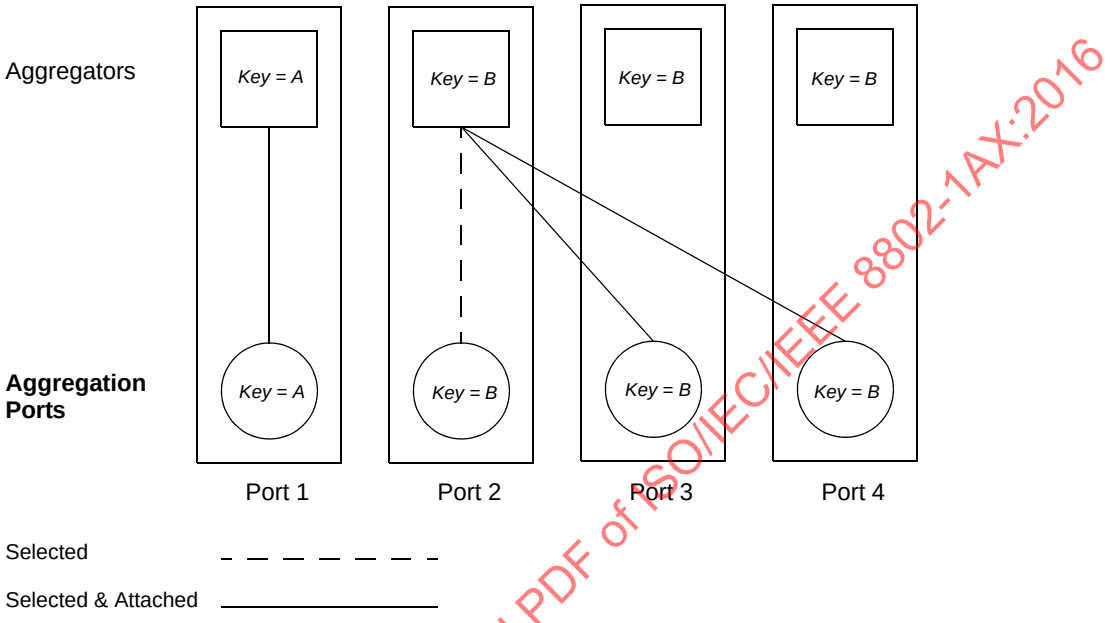


Figure 6-20—Selection of Aggregators

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

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

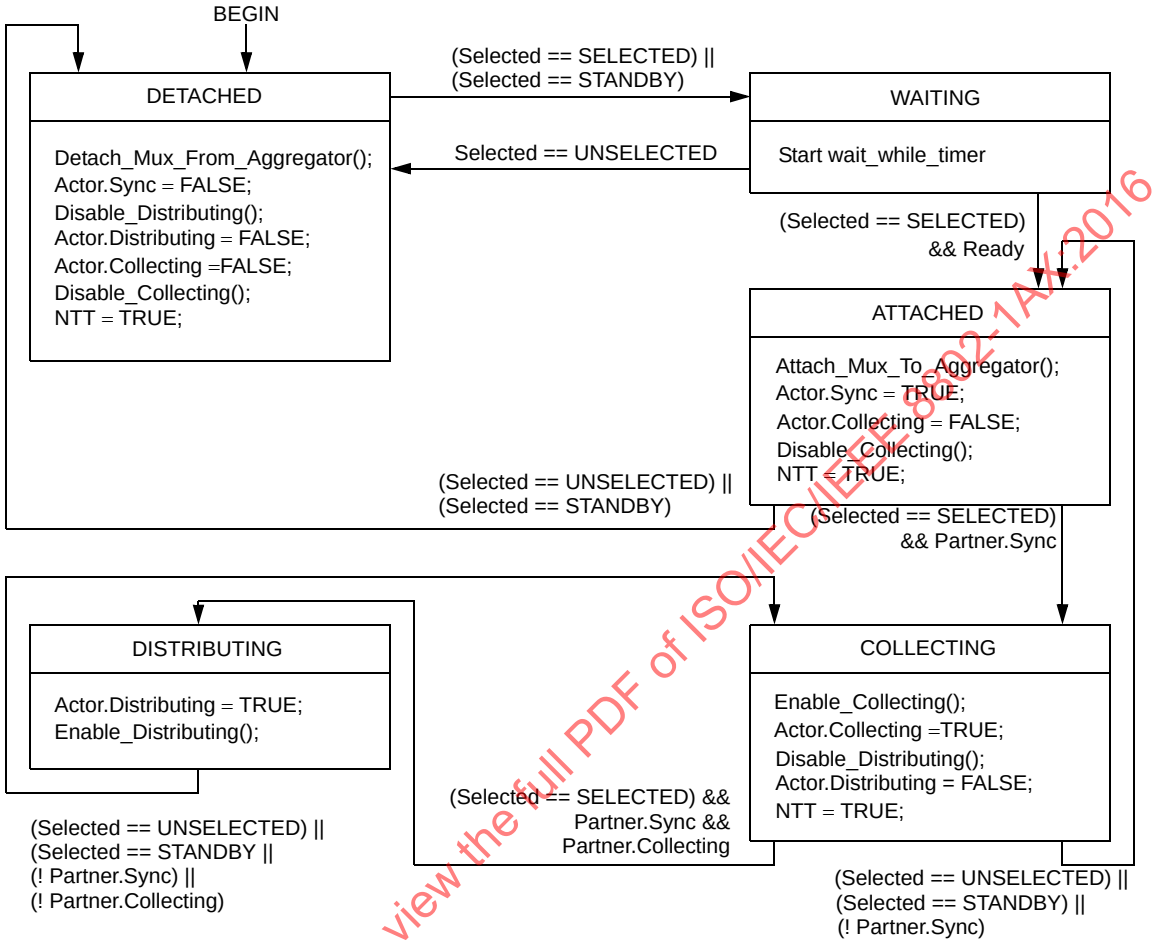
6.4.15 Mux machine

The Mux machine shall implement the function specified in either of the Mux machine state diagrams, Figure 6-21 and Figure 6-22, with their associated parameters (6.4.4 through 6.4.11).

The state machine conventions in IEEE Std 802.1Q-2014 require that all in-state actions are performed in sequence and are atomic, completing before the evaluation of any exit conditions and before the execution of procedures or evaluation of exit conditions for any other state or state machine.

The independent control state diagram (Figure 6-21) is suitable for use implementations in which it is possible to control enabling and disabling of frame collection from an Aggregation Port, and frame distribution to an Aggregation Port, independently. The coupled control state diagram (Figure 6-22) is

suitable for use implementations where collecting and distributing cannot be controlled independently with respect to an Aggregation 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 6-21—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 Key, the Partner_State.Aggregation, the Actor System ID, 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 deselecting 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 Aggregation Ports may be reconfiguring at the same time. Once the wait_while_timer expires, and once the wait_while_timers of all other Aggregation Ports that are ready to attach to the same Aggregator have expired, the process of attaching the Aggregation 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 reenters 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 Aggregation Ports will attach; for example, where all other Aggregation Ports with the same operational Key are already attached to the Aggregator.

e) If Selected is STANDBY, the Aggregation 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 reenters the DETACHED state. If SELECTED becomes SELECTED, then the operation is as described in item d). The latter case allows an Aggregation 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 Aggregation 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 Aggregation Port to be IN_SYNC, and Actor_Oper_Port_State.Collecting is set to FALSE. Collection of frames from the Aggregation Port is disabled. In the coupled control state machine, Distribution of frames to the Aggregation 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 Aggregation 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 Aggregation Port to be OUT_OF_SYNC, distribution of frames to the Aggregation Port is disabled, Actor_Oper_Port_State.Distributing and Actor_Oper_Port_State.Collecting are set to FALSE, and collection of frames from the Aggregation Port is disabled. The state machine remains in the DETACHED 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 Aggregation Port is enabled, then Actor_Oper_Port_State.Collecting is set to TRUE, then distribution of frames to the Aggregation 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 Aggregation 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).

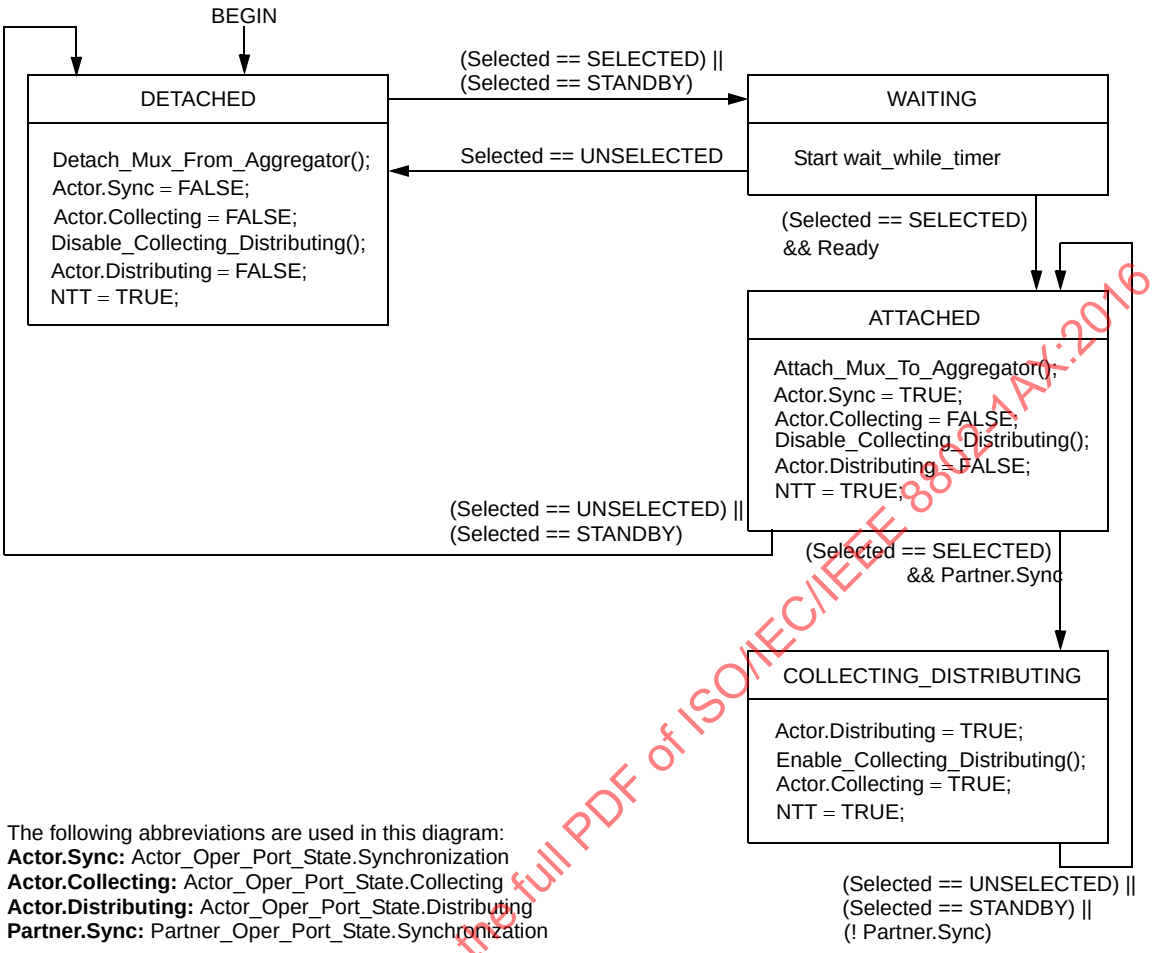


Figure 6-22—Mux machine state diagram (coupled control)

In the COLLECTING_DISTRIBUTING state, the value of Actor_Oper_Port_State.Distributing is set to TRUE, distribution of frames to the Aggregation Port and collection of frames from the Aggregation 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 an Aggregation 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 Aggregation Port into operation or taking the Aggregation 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.

6.4.16 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 Aggregation 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 (6.4.7) variable is TRUE, the Transmit machine shall ensure that a properly formatted LACPDU (6.4.2) is transmitted [i.e., issue a CtrlMuxN:M_UNITDATA.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 preceding restriction, the information sent in the LACPDU corresponds to the operational values for the Aggregation 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.

In a Version 2 implementation when enable_long_pdu_xmit and LongLACPDUtransmit are TRUE the transmitted LACPDU will be a Long LACPDU, formatted as defined in 6.4.2 and including the Port Conversation Mask TLVs (6.4.2.4.3).

6.4.17 Churn Detection machines

If implemented, the Churn Detection machines shall implement the functions specified in Figure 6-23 and Figure 6-24 with their associated parameters (6.4.4 through 6.4.11). 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 an Aggregation 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 nonstandard 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 Aggregation Ports—see 6.7.4.2) and the physical topology is such that one or more Aggregation 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 Aggregation 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 6.7 to find the ideal configuration. The churn detection timers allow 60 s 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 (6.4.4). 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 6-23.

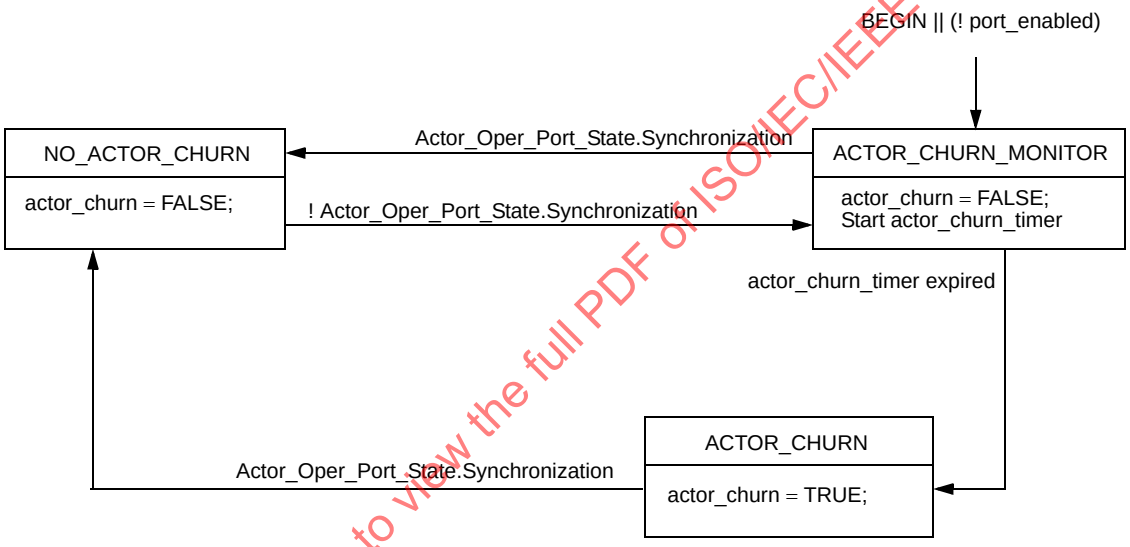


Figure 6-23—Actor Churn Detection machine state diagram

The Partner Churn Detection state machine is depicted in Figure 6-24

6.4.18 Long LACPDU machine

The Long LACPDU machine shall implement the function specified in Figure 6-25 with its associated parameters (6.4.4 through 6.4.11).

The Long LACPDU machine is only applicable to Version 2 or higher version implementations of LACP. It enables the transmission on an Aggregation Port of Long LACPDUs by an Actor implementing Version 2 or higher LACP only if the LACPDUs received on the same port report a Version 2 or higher implementation of the LACP by the Partner.

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

a) *NO_LONG_LACPDUS_SENT.* While in this state, Long LACPDUs transmission is not allowed.
b) *LONG_LACPDUS_SENT.* While in this state, Long LACPDUs transmission is allowed.

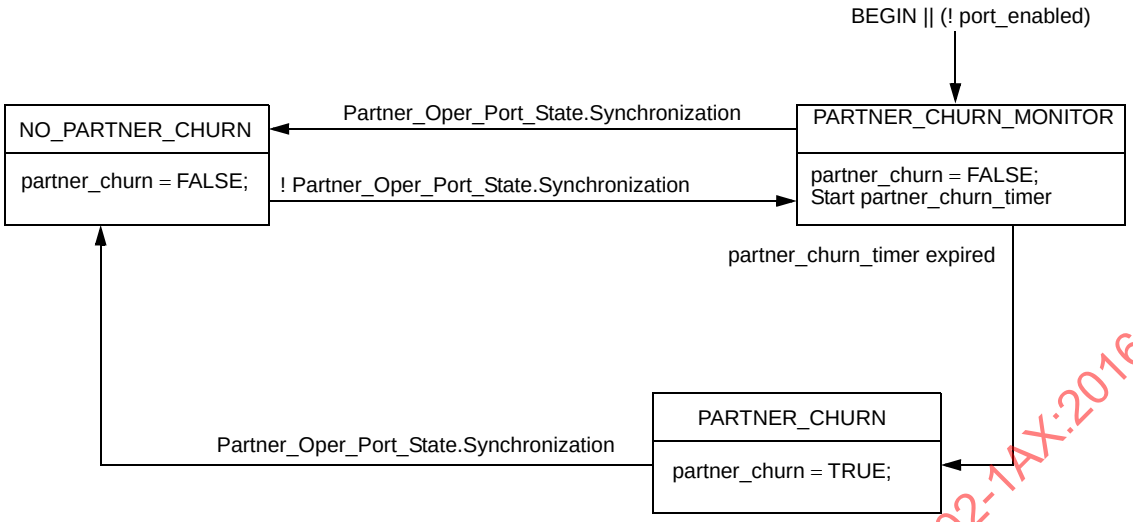


Figure 6-24—Partner Churn Detection machine state diagram

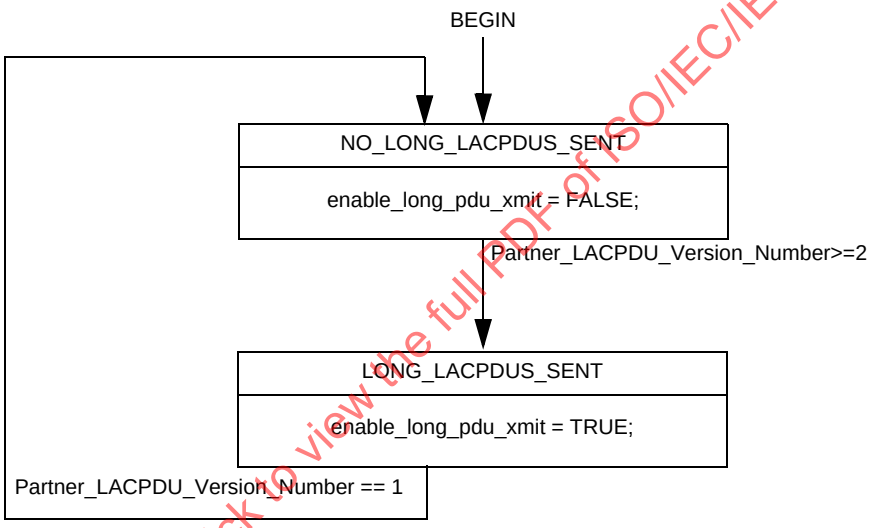


Figure 6-25—Long LACPDU machine state diagram

The System is initialized in the NO_LONG_LACPDUS_SENT and Long LACPDU transmission is prohibited. Upon reception of an LACPDU the Receive machine (6.4.12) transits to the CURRENT state and a Version 2 or higher implementation will run the recordVersionNumber function to update the Partner_LACPDU_Version_Number variable with the value in the Version Number field on the received LACPDU. If the updated value is larger or equal to 2, the System will transit to the LONG_LACPDUS_SENT state and transmission of Long LACPDUs will be enabled. The System remains in the LONG_LACPDUS_SENT state until a received LACPDU reports a Version 1 implementation by the Partner System, which triggers a state transition to the NO_LONG_LACPDUS_SENT state and the transmission of Long LACPDUs will be prohibited.

6.5 Marker protocol

6.5.1 Introduction

The Marker protocol allows the Frame 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 Frame Collection function and a Marker Response PDU is returned on the same link to the initiating Actor's Frame Distribution function. Marker and Marker Response PDUs are treated by the underlying MACs at each end of the link as normal MPDUs; 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 Frame Distribution function requests transmission of a Marker PDU on a given link and does not transmit any further MPDUs 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 MPDUs related to those conversations still to be received by the Partner's Frame Collection function. The use of the Marker protocol can therefore allow the Frame 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 Frame Collector.

NOTE—The use of the Marker protocol is further discussed in Annex B. An alternative to the Marker protocol is defined in 6.6.

The operation of the Marker protocol is unaffected by any changes in the Collecting and Distributing states associated with the Aggregation Port. Therefore, Marker and Marker Response PDUs can be sent on an Aggregation Port whose Frame Distribution function is disabled; similarly, such PDUs can be received and passed to the relevant Aggregator's Frame Collection or Distribution function on an Aggregation Port whose Frame 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 6.5.4.1 and 6.5.4.2) is mandatory. Some distribution algorithms may not require the use of a marker; other mechanisms (such as timeouts) may be used as an alternative. This standard does not specify how, when, or if the Marker protocol is used. It is possible to define a distribution algorithm that does not require the Marker Protocol, e.g., that defined in 8.2. See also Annex B.

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 have to therefore make their own provision for handling such cases.

6.5.2 Sequence of operations

Figure 6-26 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.

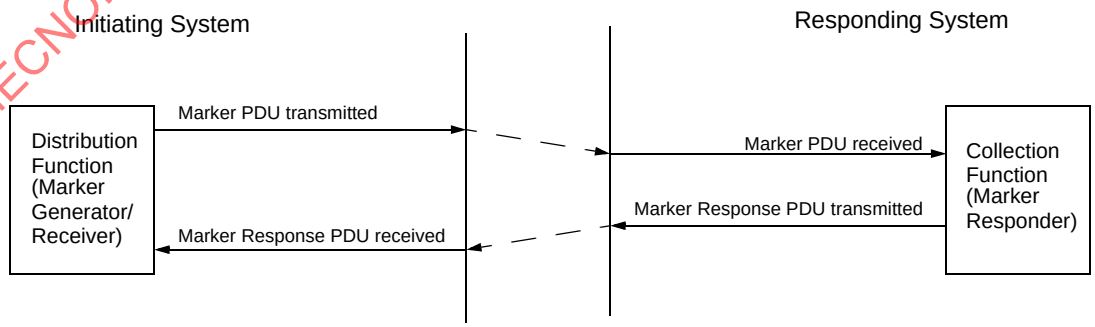


Figure 6-26—Marker protocol time sequence diagram

6.5.3 Marker and Marker Response PDU structure and encoding

6.5.3.1 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.

6.5.3.2 Encapsulation of Marker and Marker Response PDU in frames

Marker and Marker Response PDUs are encoded in the mac_service_data_unit parameter of an M_UNITDATA.request or M_UNITDATA.indication. The first octets of the mac_service_data_unit are a protocol identifier, followed by the LACPDU, followed by padding octets, if any, as required by the underlying MAC service.

Where the ISS instance used to transmit and receive frames is provided by a media access control method that can support EtherType encoding directly (e.g., is an IEEE 802.3 MAC), the protocol identifier is two octets in length, and the value is the Slow Protocols EtherType (hexadecimal 88-09).

Where the ISS instance is provided by a media access method that cannot directly support EtherType encoding (e.g., is an IEEE 802.11 MAC), the EtherType protocol identifier is encoded according to the rule for a Subnetwork Access Protocol (Clause 9 of IEEE Std 802-2014) that encapsulates Ethernet frames over LLC, and comprises the SNAP header (hexadecimal AA-AA-03) followed by the SNAP PID (hexadecimal 00-00-00) followed by the Slow Protocols EtherType (hexadecimal 88-09).

6.5.3.3 Marker and Marker Response PDU structure

The Marker PDU and Marker Response PDU structure shall be as shown in Figure 6-27 and as further described in the following field definitions:

a) *Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. Both Marker and Marker Response PDUs carry the Marker_subtype value 0x02.
b) *Version number.* This identifies the Marker protocol version; implementations conformant to this version of the standard carry the value 0x01.
c) *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.
d) *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).
e) *Requester_Port.* The Port Number assigned to the Aggregation Port by the Requester (the System sending the initial Marker PDU), encoded as an unsigned integer.

Marker PDU	Octets	Marker Response PDU
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 6-27—Marker PDU and Marker Response PDU structure

f) *Requester_System.* The MAC address component of the Requester's System ID.

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

h) *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 1—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.

i) *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.

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

k) *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 2—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.

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

6.5.4 Protocol definition

6.5.4.1 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 6-27, to the same Aggregation 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 802.3-2008, 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 1—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 802.3-2008, Annex 57A.3.

NOTE 2—The Marker Responder behaves the same whether Link Aggregation is employed in a single system or as part of a DRNI (Clause 9). In the latter case, frames in transit between one Portal System and another when a Marker Response is generated can be delivered out of order. See 6.6 for an alternative method of ensuring frame ordering.

6.5.4.2 Marker Responder state diagram

The Marker Responder shall implement the function specified in Figure 6-28, with its associated parameters (6.5.4.2.1 through 6.5.4.2.2).

6.5.4.2.1 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.request and M_UNITDATA.indication primitives.

Protocol_DA
: One of the addresses selected from Table 6-1 determined by the setting of the aAggPortProtocolDA managed object (7.3.2.2.1).
Value: 48 bits.

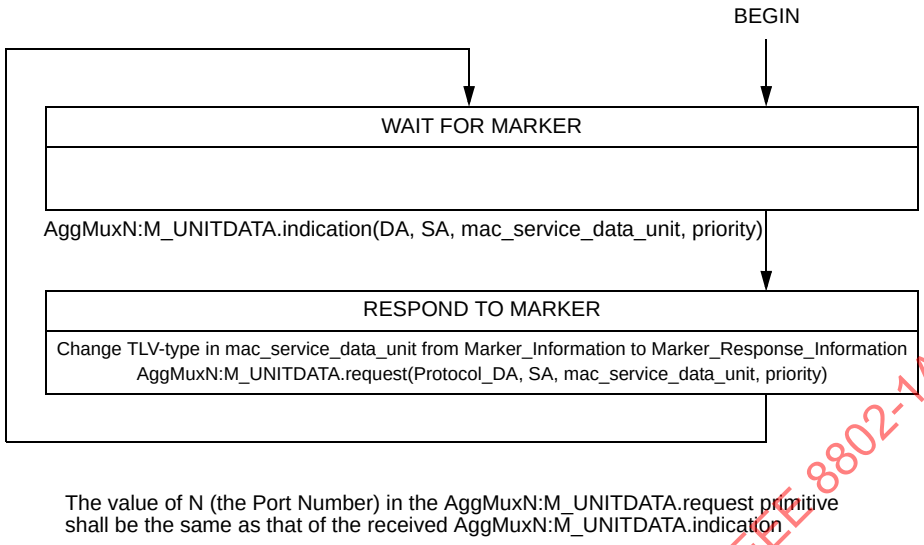


Figure 6-28—Marker Responder state diagram

6.5.4.2.2 Messages

AggMuxN:M_UNITDATA.request
: The service primitive used to transmit a frame with the specified parameters.

AggMuxN:M_UNITDATA.indication
: The service primitive used to pass a received frame to a client with the specified parameters.

Upon receipt of an AggMuxN:M_UNITDATA.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 6.5.3.3 item k), allow Version 1 devices to be compatible with future revisions of the protocol.

6.6 Conversation-sensitive frame collection and distribution

Conversation-sensitive frame collection and distribution provides a means for both the Frame Distributor and the Frame Collector to determine which Aggregation Link is to be selected by the Distributor for any given frame, and for the Collector to only accept frames received on the expected Aggregation Link. This serves a dual function. It enables the use of the same Aggregation Link for both directions of a bidirectional conversation, and it provides a means of ensuring that frames are not misordered when a conversation is moved between Aggregation Links without requiring the use of the Marker Protocol (6.5).

The use of the Marker protocol has the advantage that a single, simple Frame Collector can be used to receive frames from any Frame Distributor, no matter what distribution algorithm is used. However, ensuring that a Marker Response PDU works properly in a DRNI (Clause 9), where either or both ends of the LAG can comprise multiple Systems, becomes problematical.

A Frame Distributor and Frame Collector at the two ends of a LAG can implement Conversation-sensitive frame collection and distribution. This includes the following elements:

a) TLVs are included in LACPDUs to indicate and acknowledge the distribution algorithm in use by the Frame Distributor.
b) The Frame Distributor assigns every frame to a Port Conversation ID (8.1) and uses that Port Conversation ID to assign the frame to an Aggregation Port.
c) The Frame Collector knows and can execute the same frame-to-Conversation ID mapping that is used by the Frame Distributor.
d) TLVs included in LACPDUs allow the Frame Distributor to specify which Conversation IDs are being transmitted on each Aggregation Port. The Frame Collector stores this information in the Collection_Conversation_Mask.
e) The Frame Collector discards frames received on Aggregation Ports whose Conversation IDs are not in the list supplied by the Frame Distributor for that Aggregation Port.

Thus, frame ordering is guaranteed by the fact that the Frame Collector receives a given conversation on only one Aggregation Port. In the case of a DRNI (Clause 9), the Frame Collectors in a Portal are configured to ensure that a given conversation is received from only one Aggregation Port.

NOTE—Whether Conversation-sensitive frame collection and distribution is required of a given conformant System depends on the support of Per-service frame distribution algorithms by that System. (See Clause 8.)

The Frame Collector extracts a frame's Port Conversation ID according to the distribution algorithm employed (e.g., that specified in 8.2), then examines the Collection_Conversation_Mask variable () of the Aggregation Port on which the frame was received. The frame is passed on to the Client Port only if Collection_Conversation_Mask passes its Port Conversation ID.

6.6.1 Conversation-sensitive collection and distribution state diagrams

When a System implements Conversation-sensitive collection and distribution:

a) The Frame Distributor shall operate as specified in 6.2.4 and implement the state diagram shown in Figure 6-4 and the associated definitions contained in 6.2.4.1, constrained to use only the frame distribution, configured in the aAggPortAlgorithm (7.3.1.1.33) and the aAggConversationAdminLink[] (7.3.1.1.35), and provided by Port_Oper_Conversation_Mask () in the case that the Conversation-sensitive LACP operation (6.6.2) is supported.
b) The Frame Collector shall implement the function specified in the Conversation-sensitive collection state diagram shown in Figure 6-29 and the associated definitions contained in 6.6.1.1.

6.6.1.1 Conversion-sensitive collection state diagram

6.6.1.1.1 Variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

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

Discard_Wrong_Conversation
: Discard frames with incorrect Port Conversation IDs flag. If TRUE, the Frame Collector discards any frame received from an Aggregation Port with a Port Conversation ID not set in Collection_Conversation_Mask, and the Frame Distributor uses only the frame distribution algorithm selected by the aAggPortAlgorithm (7.3.1.1.33). If FALSE, the Frame Collector does

not filter frames received from the Aggregation Ports. The variable is set equal to aAggAdminDiscardWrongConversation (7.3.1.1.37).
Value: Boolean

6.6.1.1.2 Variables associated with each Aggregation Port

Collection_Conversation_Mask
The operational Boolean vector, indexed by Port Conversation ID, indicating whether the indexed Port Conversation ID is allowed to reach the Aggregator when received through this Aggregation Port (TRUE = passes). This variable is constructed from the agreed Conversation_PortList[], and if supported, the operation of the Conversation-sensitive LACP Link Aggregation Control Protocol (6.6.2).
Value: sequence of Boolean values, indexed by Port Conversation ID.

6.6.1.1.3 Functions

DeterminePortConversationID
This function determines a Port Conversation ID for the frame. Any algorithm can be used to determine the Port Conversation ID provided it depends only on information found within the frame and the same algorithm is used by both the Frame Distributor and Frame Collector. This function may involve the application of local Service ID to Conversation ID mappings, provided by 7.3.1.1.38, in cases where additional local information is needed to determine the Port Conversation ID, as is the case of the Per I-SID service distribution (8.2.3).

6.6.1.1.4 Messages

Agg:M_UNITDATA.indication
AggMuxN:M_UNITDATA.indication
The service primitives used to pass a received frame to a client with the specified parameters.

6.6.1.1.5 State diagram

The Conversion-sensitive collection state diagram shall implement the function specified in Figure 6-29 with its associated parameters (6.6.1.1.1 through 6.6.1.1.4).

The operation of the Frame Collector when Conversation-sensitive collection and distribution is implemented is identical to that specified in 6.2.3 when the variable Discard_Wrong_Conversation is FALSE. When Discard_Wrong_Conversation is TRUE, indicating that the Conversation-sensitive collection and distribution function is operational, the Frame Collector operation is modified so that the Frame Collector goes into the PORT_CONVERSATION_ID_CHECK state and the DeterminePortConversationID function is run to determine a Port Conversation ID for the frame. This value is used as an index to the Collection_Conversation_Mask variable and if the associated identified Boolean is TRUE the frame passes to the Aggregator Client, otherwise it gets discarded.

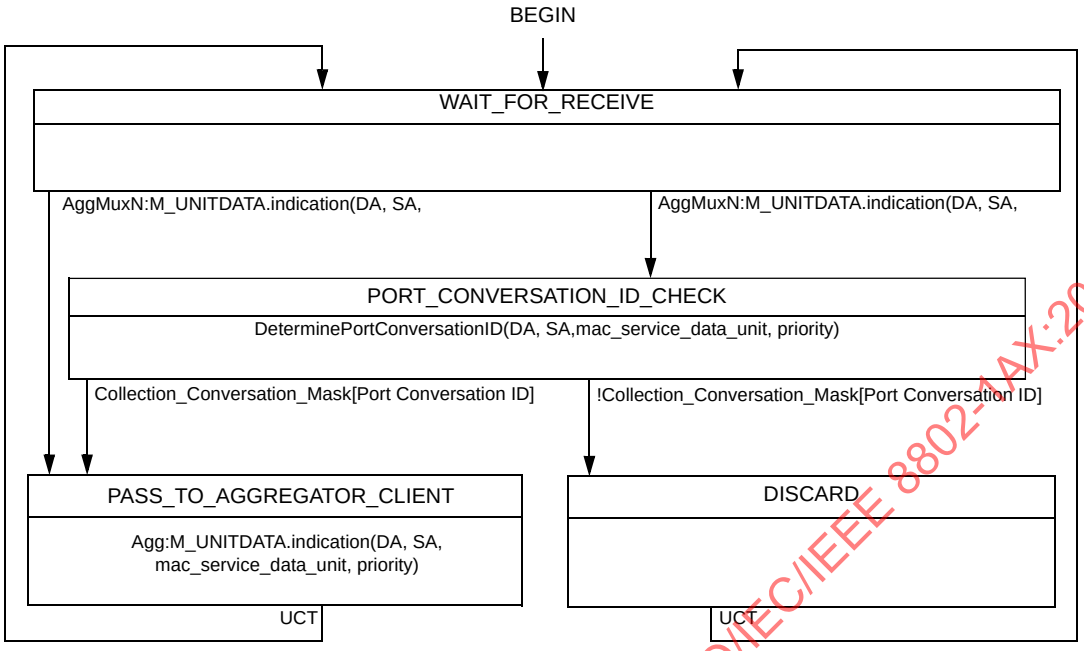


Figure 6-29—Conversation-sensitive collection state diagram

6.6.2 Conversation-sensitive LACP state diagrams

The Conversation-sensitive LACP Version 2 additions to the Version 1 LACP specified in 6.4 enable configuration, control and coordination of the Frame Collection and the Frame Distributor functions of each participating System or Portal by using static information local to each System or Portal and dynamic information exchanged by means of the Version 2 LACPDU exchanges specified in this clause.

For each Aggregation Port in a System or Portal, Conversation-sensitive LACP:

a) Maintains the agreed configuration information;
b) Exchanges configuration information with the Partner System to verify agreed per Port Conversation ID configuration choices;
c) Uses information from the Partner System's Link Aggregation Control entity to enable or disable the Aggregator's Frame Collector and Frame Distributor on a Port Conversation ID basis.

The operation of Conversation-sensitive LACP is described in detail in 6.6.2.1 through 6.6.2.7. The Conversation-sensitive LACP protocol entity is optional protocol sub-entity within the Link Aggregation Control in Figure 6-2.

6.6.2.1 Per-Aggregator Variables

Actor_Conversation_LinkList_Digest
A digest of aAggConversationAdminLink[] (7.3.1.1.35) for exchange with the Partner System. The digest, is a 16-octet MD5 fingerprint [see IETF RFC 1321 (1992)] created from the aAggConversationAdminLink[]. To calculate the digest, aAggConversationAdminLink[] is considered to contain 4096 consecutive elements, where each element contains a list of Link Number IDs, encoded as binary numbers in the order of preference, highest to lowest, followed by the Port Conversation ID. The first element of the table contains the prioritized list of Link Number IDs of Aggregation Ports assigned to Port Conversation ID 0, the second element the

prioritized list of Link Number IDs of Aggregation Ports assigned to Port Conversation ID 1, the third element the prioritized list of Link Number IDs of Aggregation Ports assigned to Port Conversation ID 2, and so on, with the last element containing the prioritized list of Link Number IDs of Aggregation Ports assigned to Port Conversation ID 4095. This variable is referenced by the Port Conversation ID Digest TLV (6.4.2.4.2).

Value: MD5 digest

Actor_Conversation_Service_Mapping_Digest

A digest of aAggAdminServiceConversationMap[] for exchange with the Partner System. The digest, is a 16-octet MD5 fingerprint [see IETF RFC 1321 (1992)] created from the aAggAdminServiceConversationMap[]. To calculate the digest, aAggAdminServiceConversationMap[] is considered to contain 4096 consecutive elements, where each element contains, in general a set of Service IDs (8.2.2) encoded as binary numbers in increasing order followed by the Port Conversation ID to which they are mapped. If the Service IDs are representing VIDs, only a single 12-bit VID is mapped to each service, while in the case that Service IDs are representing I-SIDs, more than one 24-bit I-SIDs are possible. The first element of the table contains the Service ID assigned to Port Conversation ID 0, the second element the Service ID assigned to Port Conversation ID 1, the third element the Service ID assigned to Port Conversation ID 2, and so on, with the last element containing the Service ID assigned to Port Conversation ID 4095. This variable is referenced by the optional Port Conversation Service Mapping TLV (6.4.2.4.4) when the Service IDs are different from the Port Conversation IDs and a table is associating their values.

Value: MD5 digest

Actor_Port_Algorithm

The algorithm used by the Actor to assign frames to Port Conversation IDs. Always set equal to aAggPortAlgorithm (7.3.1.1.33). This variable is referenced by the Port Algorithm TLV (6.4.2.4.1)

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

Conversation_PortList[]

This array holds the mapping of Port Conversation IDs to Aggregation Ports inside the System or Portal, derived from the primary aAggConversationAdminLink[] managed object. An array of 4096 lists, indexed by Conversation ID, that determines which Aggregation Port in this System (or Portal) carries which Conversation ID. Each item in the array is a list of the Port IDs (6.3.4) of the Aggregation Ports in this System (or in this Portal, if this Aggregator is attached to a DR Function [9.3]), in priority order from most desired to least desired, for carrying the indexed Conversation ID. This variable is updated by the updateConvesationPortList function, which is always invoked when a new aAggConversationAdminLink[] (7.3.1.1.35) or new aAggPortLinkNumberID (7.3.2.1.27) operator command is issued.

Value: sequence of Port IDs (6.3.4)

Differ_Conversation_Service_Digests

A Boolean indicating that the Conversation Service Mapping Digests used by the Systems or Portals at the two ends of the LAG are different.

Value: Boolean

Differ_Port_Algorithms

A Boolean indicating that the Port Algorithms used by the Systems or Portals at the two ends of the LAG are different.

Value: Boolean

Differ_Port_Conversation_Digests

A Boolean indicating that the Port Conversation Digests used by the Systems or Portals at the two ends of the LAG are different.

Value: Boolean

Partner_Admin_Conversation_PortList_Digest

The value for the digest of the prioritized Port Conversation ID-to-Link Number ID assignments of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Its default value is set to NULL.
Value: MD5 digest

Partner_Admin_Conversation_Service_Mapping_Digest
The value for the digest of the Port Conversation ID-to-Service ID assignments of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Its default value is set to NULL.
Value: MD5 digest

Partner_Admin_Port_Algorithm
The value for the algorithm of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Its default value is set to NULL.
Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

Partner_Conversation_PortList_Digest
The last-received digest of the Partner System's prioritized Port Conversation ID-to-Aggregation Port assignments.
Value: MD5 digest

Partner_Conversation_Service_Mapping_Digest
The last-received digest of the Partner System's Port Conversation ID-to-Service ID assignments.
Value: MD5 digest

Partner_Port_Algorithm
The last-received value of the algorithm used by the Partner System to assign frames to Port Conversation IDs.
Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

6.6.2.2 Variables associated with each Aggregation Port

Admin_Link_Number_ID
The Link Number ID value configured for this Aggregation Port by the System's administrator. It is always set equal to aAggPortLinkNumberID (7.3.2.1.27).
Value: Integer, 0 to 65535

ChangeActorOperDist
This variable tracks the variable Actor_Oper_Port_State.Distributing. It sets the WTR_timer to aAggPortWTRTime when the Actor_Oper_Port_State.Distributing changes from FALSE to TRUE and there is at least one Aggregation Port in the same LAG having a non NULL Port_Oper_Conversation_Mask. The variable is set to TRUE when the WTR_timer is running and the Actor_Oper_Port_State.Distributing changes from TRUE to FALSE or when the WTR_timer expires and Actor_Oper_Port_State.Distributing is TRUE. This variable is also set to TRUE if new values for the aAggConversationAdminLink[] (7.3.1.1.35) or aAggAdminServiceConversationMap[] (7.3.1.1.38) are initiated. When the variable is set to TRUE, updateLocal is set to TRUE on all other Aggregation Ports attached to the same Aggregator. ChangeActorOperDist is set to FALSE by the updateConversationMask function.
Value: Boolean

ActPar_Sync
This variable indicates if the Conversation Mask used by the Actor's Frame Distributor is the same or not as that used by the Partner Frame Distributor in the same System. TRUE if Partner_Oper_Conversation_Mask == Port_Oper_Conversation_Mask, otherwise FALSE. This variable is referenced by the Port Conversation Mask-1 TLV (6.4.2.4.3).
Value: Boolean

Comp_Oper_Conversation_Mask
: The operational value of the Port_Oper_Conversation_Mask Boolean vector of the System or Portal as computed by updateConversationMask.
: Value: sequence of Boolean values, indexed by Port Conversation ID.

Link_Number_ID
: This System's operational Link Number ID value for this Aggregation Port. It is used to assign the Aggregation Port to the Link Number ID, which is an identifier commonly used by this System's administrator and the Partner System's administrator to refer to the interconnected link.
: Value: Integer, 0 to 65535

LongLACPDUtransmit
: This variable indicates to the Transmit machine (6.4.16) that it needs to transmit Long LACPDUs including the Port Conversation Mask TLVs (6.4.2.4.3). It is set to TRUE while the current_while_Long_LACP_timer runs and set to FALSE otherwise.
: Value: Boolean

NTT
: Need To Transmit flag. This variable is reused from the basic LACP operation (6.4.7).
: 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.

Partner_Admin_Conversation_Mask
: The Port_Oper_Conversation_Mask Boolean vector of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Its default is set to NULL.
: Value: sequence of Boolean values, indexed by Port Conversation ID.

Partner_Admin_Link_Number_ID
: The value for the Link Number ID of the Partner System for this Aggregation Port, assigned by administrator or System policy for use when the Partner's information is unknown. It is always equal to aAggPortPartnerAdminLinkNumberID (7.3.2.1.28).
: Value: Integer, 0 to 65535.

Partner_ActPar_Sync
: The last-received value of the Partner System's ActPar_Sync value.
: Value: Boolean

Partner_DWC
: The last-received value of the Partner System's DWC value.
: Value: Boolean

Partner_Link_Number_ID
: The last-received value of the Partner System's Link Number ID value.
: Value: Integer, 0 to 65535

Partner_Oper_Conversation_Mask
: The last received Boolean vector indicating whether the indexed Port Conversation ID can be distributed through the Partner Systems' Aggregation Port on the link (TRUE = passes).
: Value: sequence of Boolean values, indexed by Port Conversation ID.

Partner_PSI
: The last-received value of the Partner System's PSI value.
: Value: Boolean

Port_Oper_Conversation_Mask
: The operational Boolean vector, indexed by Port Conversation ID, indicating whether the indexed Port Conversation ID is distributed through this Aggregation Port (TRUE = passes). This variable is constructed from the agreed Conversation_PortList[] and the operation of the Conversation-sensitive LACP Link Aggregation Control Protocol (6.6.2). This variable is split in four parts, Port_Oper_Conversation_Mask_1, Port_Oper_Conversation_Mask_2,

Port_Oper_Conversation_Mask_3, and Port_Oper_Conversation_Mask_4 in order to be carried by the Port Conversation Mask TLVs as specified in 6.4.2.4.3.
Value: sequence of Boolean values, indexed by Port Conversation ID.

updateLocal
This variable is set to TRUE by the recordReceivedConversationMaskTLV function when the value of either ActPar_Sync or Partner_ActPar_Sync is FALSE.

6.6.2.3 Variables used for managing the operation of the state diagrams

BEGIN
This variable indicates the initialization (or reinitialization) of the Conversation-sensitive 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

actor_CDS_churn
This variable indicates that the Actor CDS Churn Detection machine has detected that the Conversation Masks used by Frame Distributor and the Frame Collector in the local System have failed to converge within a specified time, and that management intervention is required.
Value: Boolean

ChangeAggregationPorts
This variable tracks the operational state of all Aggregation Ports associated to this System and is set to TRUE when any of them changes. It is the logical OR of the ChangeActorOperDist variables for all Aggregation Ports.
Value: Boolean

partner_CDS_churn
This variable indicates that the Partner CDS Churn Detection machine has detected that the Conversation Masks used by Frame Distributor and the Frame Collector in the remote System have failed to converge within a specified time, and that management intervention is required.
Value: Boolean

updateAllLocal
This variable is the logical OR of the updateLocal variables for all Aggregation Ports.
Value: Boolean

6.6.2.4 Functions

recordReceivedConversationMaskTLV
This function records the parameter value for the *ActPar_Sync* carried in a received Port Conversation Mask-1 TLV (6.4.2.4.3) as the current operational parameter values for the Partner_ActPar_Sync, it concatenates the value of *Port_Oper_Conversation_Mask_1*, *Port_Oper_Conversation_Mask_2*, *Port_Oper_Conversation_Mask_3*, and *Port_Oper_Conversation_Mask_4* carried by the Port Conversation Mask-1 TLV, ..., etc. (6.4.2.4.3), and it records the concatenation as the current value for the Partner_Oper_Conversation_Mask variable. If this function is implemented by a Portal System (Clause 9), it also records the parameter value for the *PSI* carried in a received Port Conversation Mask-1 TLV (6.4.2.4.3) as the current operational parameter value for the Partner_PSI.

It then compares the variable Port_Oper_Conversation_Mask to the Partner_Oper_Conversation_Mask and, if they are different:
It sets ActPar_Sync to FALSE;
It sets updateLocal to TRUE;
Else:
It sets ActPar_Sync to TRUE;
If Partner_ActPar_Sync is FALSE, sets updateLocal to TRUE.

recordConversationPortDigestTLV

This function records the parameter value for the *Link_Number_ID* and the *Actor_Conversation_LinkList_Digest* carried in a received Port Conversation ID Digest TLV (6.4.2.4.2) as the current operational parameter value for the Partner_Link_Number_ID and the Partner_Conversation_PortList_Digest, respectively.

It compares the newly updated value of the Partner_Link_Number_ID to the value of Link_Number_ID and if they are different, then:

The reportToManagement function is invoked to alert the Manager to the existence of a configuration error in the Aggregation ports selection choices for the Port Conversation IDs between the Systems or Portals at the ends of the LAG. In addition if this System's or Portal's Identifier is numerically larger than the Partner's, then

Link_Number_ID == Partner_Link_Number_ID; and

The function updateConvesationPortList is invoked.

It then compares the newly updated value of the Partner_Conversation_PortList_Digest to the value of Actor_Conversation_LinkList_Digest and, if they are different:

The variable Differ_Port_Conversation_Digests is set to TRUE.

Otherwise:

The variable Differ_Port_Conversation_Digests is set to FALSE.

recordConversationServiceMappingDigestTLV

This function records the parameter value for the *Actor_Conversation_Service_Mapping_Digest* carried in a received Port Conversation Service Mapping TLV (6.4.2.4.4) as the current operational parameter value for Partner_Conversation_Service_Mapping_Digest.

It then compares the newly updated value of the Partner_Conversation_Service_Mapping_Digest to the value of Actor_Conversation_Service_Mapping_Digest and, if they are different:

The variable Differ_Conversation_Service_Digests is set to TRUE.

Otherwise:

The variable Differ_Conversation_Service_Digests is set to FALSE.

recordDefaultConversationMask

This function sets Partner_Oper_Conversation_Mask and Collection_Conversation_Mask to Partner_Admin_Conversation_Mask and sets ActPar_Sync to FALSE.

recordDefaultConversationPortDigest

This function records the default parameter values for the Partner as follows:

Partner_Conversation_PortList_Digest
== Partner_Admin_Conversation_PortList_Digest;

Partner_Link_Number_ID == Partner_Admin_Link_Number_ID.

It also initializes Link_Number_ID to its configured value as follows:

Link_Number_ID == Admin_Link_Number_ID

recordDefaultConversationServiceMappingDigest

This function records the default parameter value for the Partner provided by the Partner_Admin_Conversation_Service_Mapping_Digest as the current Partner operational parameter value for Partner_Conversation_Service_Mapping_Digest.

recordDefaultPortAlgorithm

This function records the default parameter value for the Partner provided by the Partner_Admin_Port_Algorithm as the current Partner operational parameter value for Partner_Port_Algorithm.

recordPortAlgorithmTLV

This function records the parameter value for the *Actor_Port_Algorithm* carried in a received Port Algorithm TLV (6.4.2.4.1) as the current operational parameter value for Partner_Port_Algorithm.

It then compares the newly updated value of the Partner_Port_Algorithm to the value of Actor_Port_Algorithm and if they are different or any of them has the value 0 (Table 6-4):

The variable Differ_Port_Algorithms is set to TRUE.

Otherwise:

The variable Differ_Port_Algorithms is set to FALSE.

reportToManagement

To alert the Manager to the existence of a configuration error in the algorithm or agreed Aggregation ports selection choices for the Port Conversation IDs between the Systems or Portals at the ends of the LAG.

resetAllChangeActorOperDist

To reset ChangeActorOperDist to FALSE on all ports.

resetAllupdateLocal

To reset updateLocal to FALSE on all ports.

updateConversationMask

This function computes a new value for the Port_Oper_Conversation_Mask based on the Conversation_PortList[].

It first computes a new Boolean vector for the Comp_Oper_Conversation_Mask to be used by this Aggregation Port as follows: For every indexed Port Conversation ID, the selection priority list for all Aggregation Ports in the same LAG, which is provided by Conversation_PortList[], is checked to identify and exclude all Aggregation Ports for which the Actor_Oper_Port_State.Distributing = FALSE [condition that covers the cases where the associated Aggregation Ports are either non operable (port_enabled = FALSE), or in an EXPIRED state, or not in the LAG] and all Aggregation Ports for which the WTR_timer is running. The new Boolean vector for the Comp_Oper_Conversation_Mask to be used by this Aggregation Port is computed based on this modified priority list by setting TRUE for every indexed Port Conversation ID that has the highest selection priority for this Aggregation Port. If this function is implemented by a Portal System (Clause 9), with its DRF_Portal_System_Number value set to a value that is different than 1, its Portal's System Identifier set to a value that is numerically lower than the Partner's System Identifier, and PSI == Partner_PSI == TRUE, then Comp_Oper_Conversation_Mask is set to NULL. Such functionality is set to prohibit network partition when the Portal Systems in both interconnected Portals become isolated.

It then compares the value of Port_Oper_Conversation_Mask to the new value of Comp_Oper_Conversation_Mask. If they are different:

It updates Collection_Conversation_Mask as follows:

If ChangeAggregationPorts == TRUE or, updateAllLocal == TRUE, or, when the function is implemented by a Portal System, IppAllUpdate == TRUE;

It sets the Collection_Conversation_Mask to the Boolean vector corresponding to the result of the logical AND operation between the Comp_Oper_Conversation_Mask and the current Collection_Conversation_Mask;

Otherwise if ChangeAggregationPorts == FALSE and updateAllLocal == TRUE and, when the function is implemented by a Portal System, IppAllUpdate == FALSE;

It sets Collection_Conversation_Mask = Comp_Oper_Conversation_Mask;

It updates the Port_Oper_Conversation_Mask to the new value of Comp_Oper_Conversation_Mask;

Otherwise:

it leaves Port_Oper_Conversation_Mask and Collection_Conversation_Mask unmodified.

Finally it sets ChangeActorOperDist to FALSE and if ActPar_Sync or Partner_ActPar_Sync is FALSE it starts the current_while_Long_LACP_timer, using Slow_Periodic_Time (6.4.4) as

the start value; sets NTT to TRUE; and keeps the variable LongLACPDUtransmit TRUE until the timer's expiration.

updateConvesationPortList

This function transforms the mapping of Port Conversation IDs to Link Number IDs configured by the aAggConversationAdminLink[] managed object into the equivalent mapping of Port Conversation IDs to Port IDs used inside the System or Portal. The function updates Conversation_PortList[] based on the current aAggConversationAdminLink[] and the operational Link_Number_ID values. The resultant Conversation_PortList[] provides the operational Aggregation Port agreed selection prioritized list per Port Conversation ID expressed in terms of Port IDs. The function is always invoked when a new command to change aAggConversationAdminLink[] or aAggPortLinkNumberID is issued.

6.6.2.5 Timers

current_while_Long_LACP_timer

This timer is used to indicate the transmission of Long LACPDUs. It is always started with the value Slow_Periodic_Time (6.4.4).

actor_CDS_churn_timer

This timer is used to detect Actor CDS churn states. It is started using the value Churn_Detection_Time (see 6.4.4).

partner_CDS_churn_timer

This timer is used to detect Partner CDS churn states. It is started using the value Churn_Detection_Time (see 6.4.4).

WTR_timer

This timer is used to prevent frequent distribution changes due to an intermittent defect. It allows for a fixed period of time to elapse when the Actor_Oper_Port_State.Distributing changes from FALSE to TRUE before updating the Port_Oper_Conversation_Mask. It is started using the value aAggPortWTRTime (7.3.2.1.29).There is one per Aggregation Port.

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

6.6.2.6 Messages

CtrlMuxN:M_UNITDATA.indication(LLACPDU)

This message is generated by the Control Parser as a result of the reception of a Long LACPDU, formatted as defined in 6.4.2.

6.6.2.7 State diagrams

The Conversation-sensitive LACP state diagrams shall implement the functions specified in Figure 6-31, Figure 6-32, Figure 6-33, and Figure 6-34 and may implement the functions in Figure 6-35 and Figure 6-36, with their associated parameters (6.6.2.1 through 6.6.2.6 and in 6.4). There are six Conversation-sensitive LACP state diagrams for every Aggregation Port in the LAG, as follows:

a) Verification state diagram (*VER*) depicted in Figure 6-31
b) Report for Management Action state diagram *(RMA)* depicted in Figure 6-32
c) Receive Long LACPDU state diagram (*RXL*) depicted in Figure 6-33
d) Update Mask state diagram (*UM*) depicted in Figure 6-34
e) Actor Collection and Distribution Sensitive (CDS) Churn Detection state machine depicted in Figure 6-35
f) Partner Collection and Distribution Sensitive (CDS) Churn Detection state machine depicted in Figure 6-36

Figure 6-30 illustrates the relationships among these state machines and the flow of information between them.

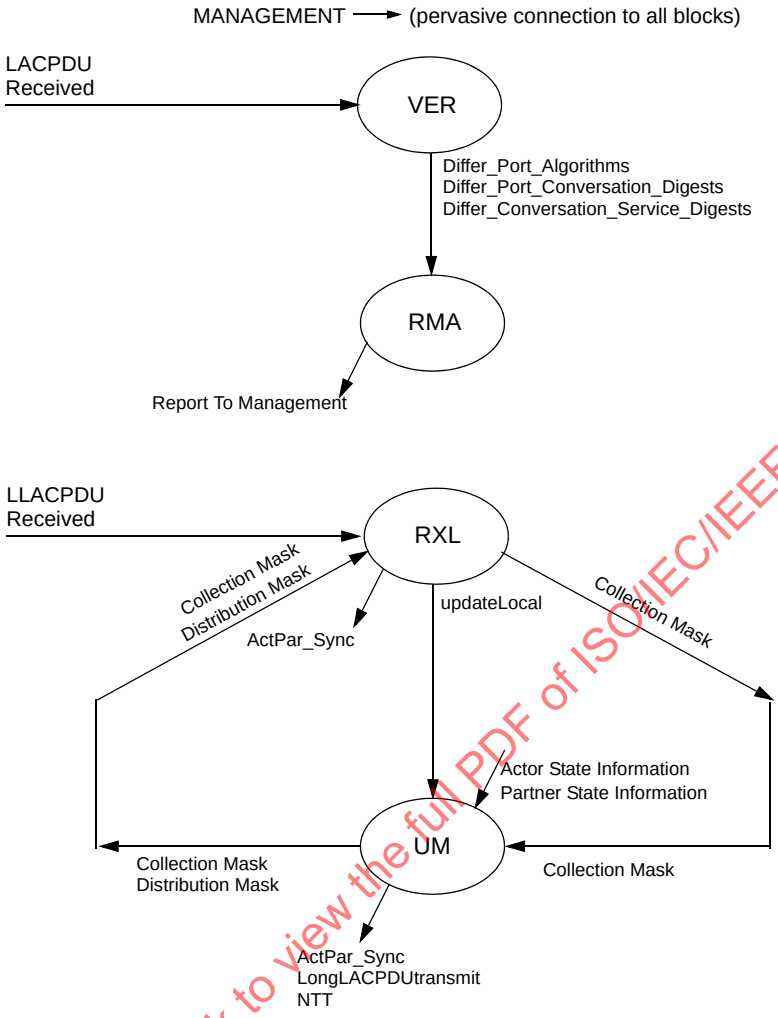


Figure 6-30—Interrelationships among Conversation-sensitive LACP state machines

The set of arrows labeled Distribution Mask represents the Port_Oper_Conversation_Mask, contained in transmitted Long LACPDUs or supplied by administrative default values, and being updated by the Update Mask machine. The set of arrows labeled Collection Mask represents the flow of updated Collection_Conversation_Mask values between the RXL and UM state machines. The set of arrows labeled Partner State Information and Actor State information are as defined in 6.4.3. Partner State Information represents new Partner information, contained in an incoming LACPDU or supplied by administrative default values, being fed by the RX machine. Actor State Information represents the flow of updated Actor state information as provided by the LACP 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.

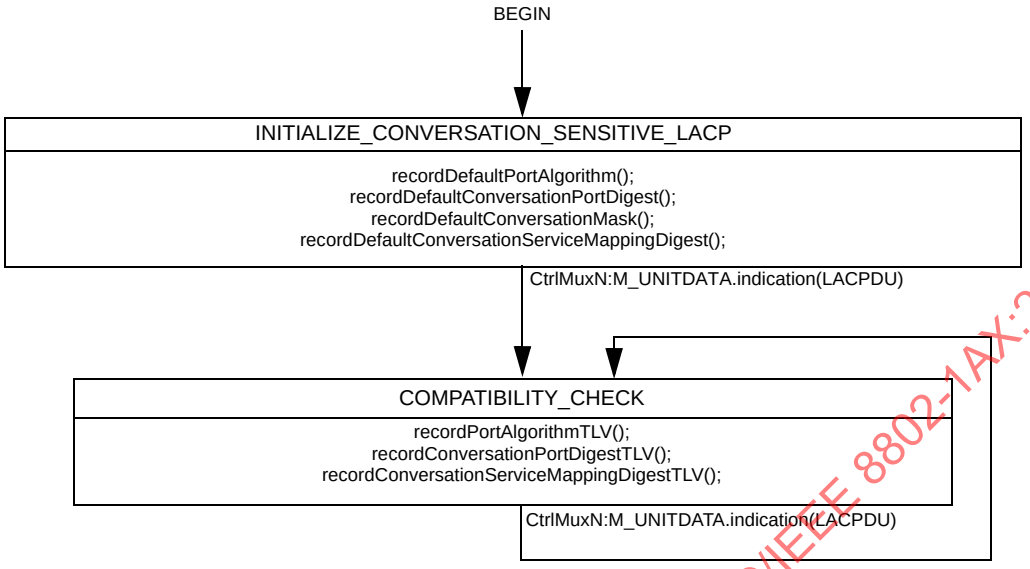


Figure 6-31—Verification state diagram

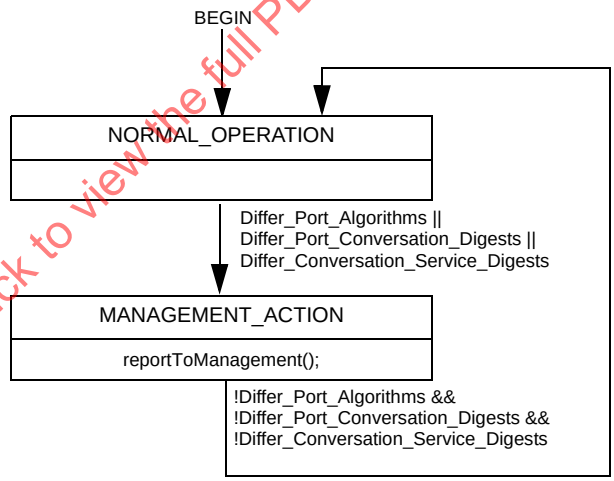


Figure 6-32—Report for Management Action state diagram

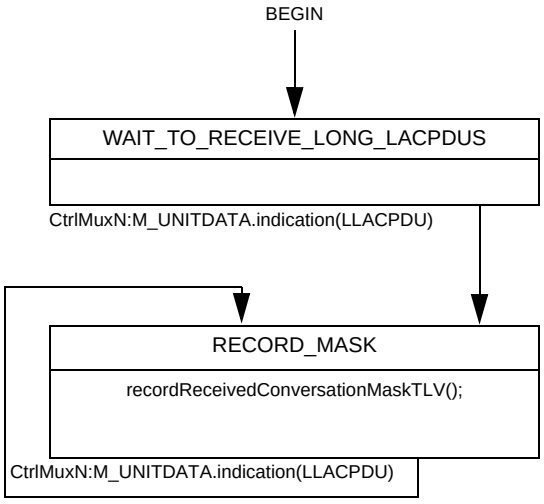


Figure 6-33—Receive Long LACPDU state diagram

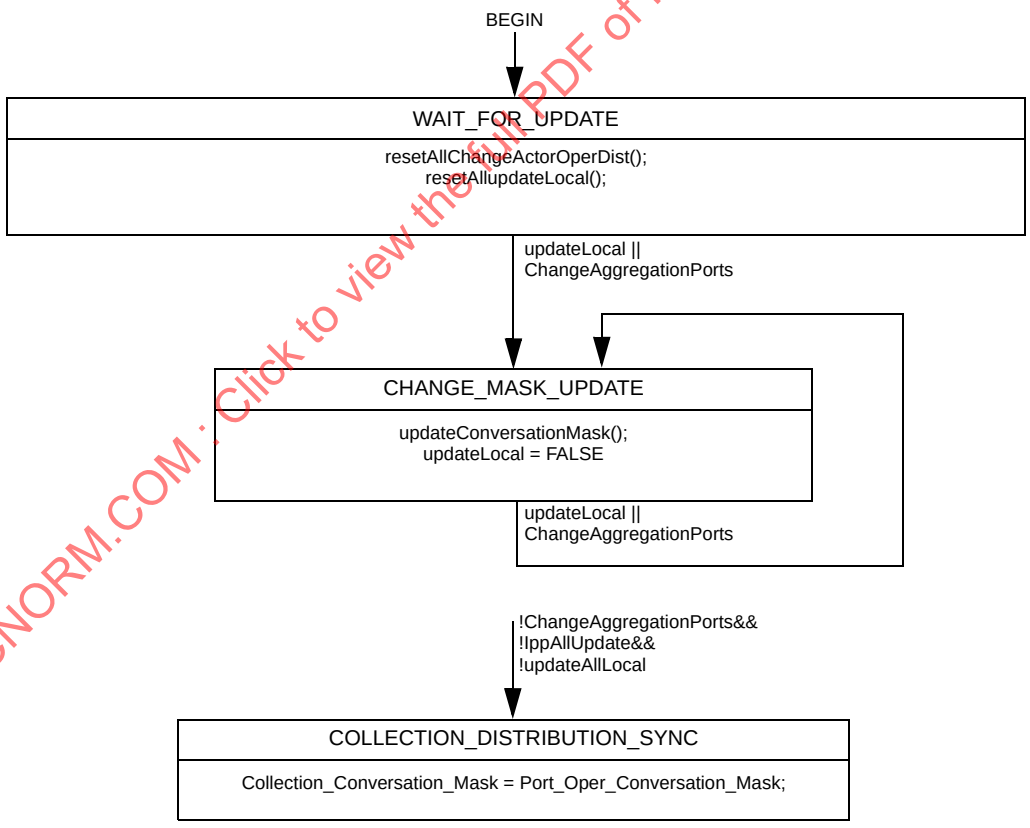


Figure 6-34—Update Mask state diagram

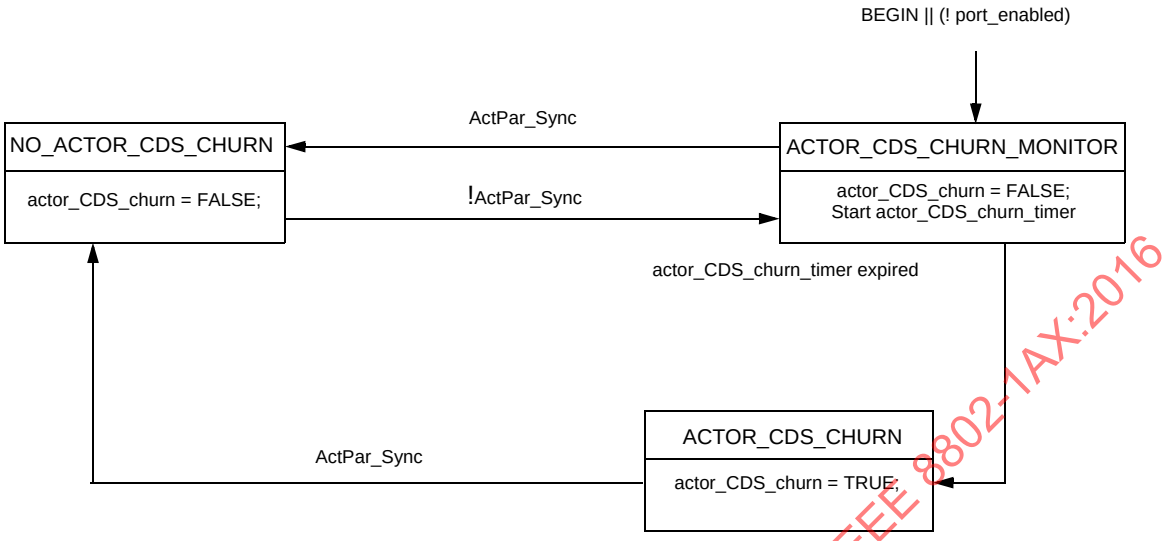


Figure 6-35— Actor CDS Churn Detection machine state diagram

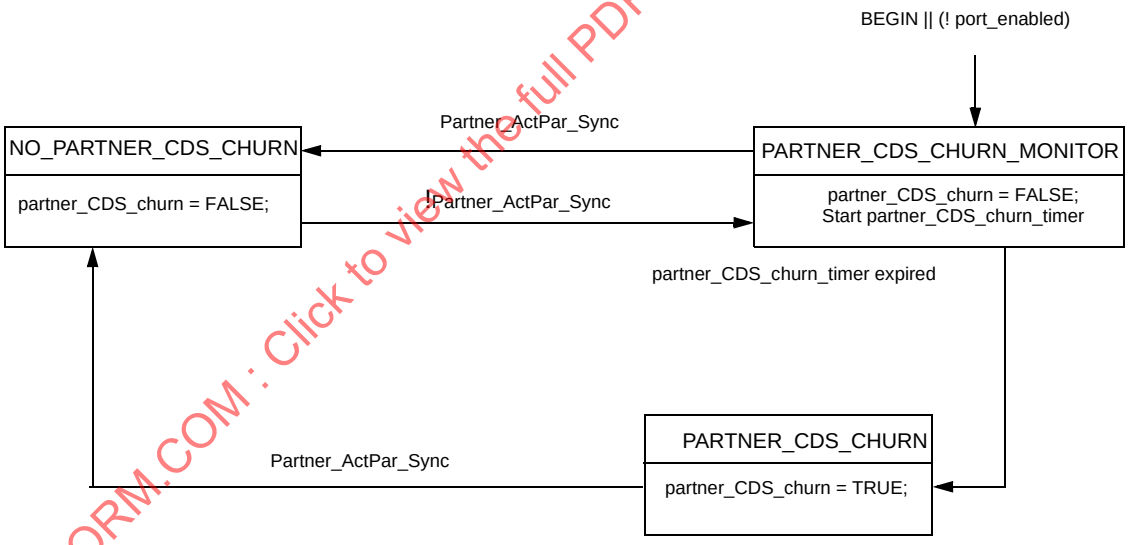


Figure 6-36—Partner CDS Churn Detection machine state diagram

These state diagrams are not required unless both Systems or Portals are running a Version 2 implementation of LACP. Through the basic LACP operation (6.4), a LAG is established between an agreed set of Aggregation Ports that are configured to use the same Operational Keys to their selected Aggregator. All Aggregation Ports in each participating System or Portal are using the same Operational Keys to their selected Aggregator. In addition, the two Systems or Portals have already agreed, by some administrative means, to use a common algorithm to assign frames to Port Conversation IDs and this information has been configured in both Systems or Portals (7.3.1.1.33). Furthermore, the two Systems or Portals have agreed on the same selection priority list of Aggregation Ports, for each Port Conversation ID, and based on this, each

System can configure itself in order to accept and transmit the agreed services on their selected Aggregation Ports as follows.

On Initialization, both Systems or Portals are configuring their Version 2 TLVs, with default values. The Port Algorithm TLV (6.4.2.4.1) is configured with the agreed algorithm (7.3.1.1.33), an MD5 digest of aAggConversationAdminLink[] (7.3.1.1.35) is used to configure the Port Conversation ID Digest TLV (6.4.2.4.2), while in cases where frames are distributed to physical links tagged with Service IDs being different than the Port Conversation IDs (8.2.3), an MD5 digest of aAggAdminServiceConversationMap[] (7.3.1.1.38) is used to configure the Port Conversation Service Mapping TLV (6.4.2.4.4).

Reception of an LACPDU when in the INITIALIZE _CONVERSATION_SENSITIVE_LACP state triggers compatibility tests. If the algorithm carried in the received Port Algorithm TLV (6.4.2.4.1), and/or the set of reported Port Conversation IDs carried in the received Port Conversation ID Digest TLV (6.4.2.4.2), and/or the Service ID to Port Conversation IDs mappings carried in the received Port Conversation Service Mapping TLV (6.4.2.4.4), are different than the associated expected configurations, the state machine transits to the MANAGEMENT_ACTION and the management system is notified in order to take further actions such as to report on the Partner System's inconsistencies, to initialize the operation of the Marker protocol, or other actions that are outside the scope of this standard.

Any event that changes one or more of the following:

— operational status of any of the Aggregation Ports in the LAG;
— configured selection rules provided by aAggConversationAdminLink[] (7.3.1.1.35);
— Service ID to Port Conversation IDs provided by aAggAdminServiceConversationMap[] (7.3.1.1.38);

will trigger a transition to the CHANGE_MASK_UPDATE state and (re-)calculation of the Conversation Masks. The updateConversationMask() computes a Distribution Conversation Mask, which is provided by the Port_Oper_Conversation_Mask variable, and a Collection Conversation Mask for the Aggregation port based on the agreed selection rules, and starts the current_while_Long_LACP_timer using Slow_Periodic_Time as the start value, and keeps the variable LongLACPDUtransmit TRUE until the timer's expiration, which causes the Transmit machine to transmit Long LACPDUs whenever an NTT trigger is set. The Collection Conversation Mask will always be set equal to the Distribution Conversation Mask when the Distribution Conversation Masks of all Aggregation Ports in the same System have been updated as a result of the local change. In the case of a Portal (Clause 9) such condition includes updates of the IPP Conversation Masks.

On receipt of a Long LACPDU the receiving Aggregation Port stores the Partner's distribution Conversation Mask carried in received Port Conversation Mask TLVs (6.4.2.4.3), and compares it with its own. If they differ, or the port on the other end of the LAG is reporting that the Conversation Masks are not in sync, further Long LACPDU exchanges will be triggered until both Aggregation Ports on the two ends of the LAG start reporting the same set of Conversation Masks. If no differences are reported, the current_while_Long_LACP_timer will terminate and the subsequent LACPDU exchanges will not include the Port Conversation Mask TLVs.

The CDS Churn Detection machines detect the situation where the Actor and Partner Conversation Masks cannot synchronize within a bounded time period. Under normal operation of the Conversation-Sensitive LACP, agreement between Actor and Partner on the operational Conversation Masks used by the Frame Distributor and the Frame Collector will be reached rapidly. Continued failure to reach agreement can be symptomatic of device failure, of the presence of nonstandard devices, or of misconfiguration. Detection of this condition is signaled by the CDS Churn Detection machines to management in order to prompt administrative action to further diagnose and correct the fault.

The Actor CDS Churn Detection state machine detects a failure of the Actor's and Partner's Conversation Mask to converge. Under normal conditions, this is ample time for convergence to take place. Similarly, the Partner Churn Detection state machine detects a failure on the Partner's view of the Conversation Masks to converge.

NOTE—Administrative control of the port_enabled variable on an Aggregation Port through its associated MAC_Enabled variable (11.2 of IEEE Std 802.1AC-2012) provides means for manual switch capabilities per Aggregation Port in conversation-sensitive LACP. The operation of a manual switch on a Port Conversation ID basis is similar to that of changing the configured priority list (7.3.1.1.34).

6.7 Configuration capabilities and restrictions

6.7.1 Use of system and port priorities

The Link Aggregation Group identifier (LAG ID) format specified by this standard cannot represent two or more LAGs that share the same combination of {SK, TL}. The use of static keys in combination with size restriction (e.g., for a Key Group containing six members, restricting the size of any aggregation to four members or fewer) leads to one LAG.

In Systems that have limited aggregation capability of this form, the following algorithm shall be used to determine the subset of Aggregation 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 MAC address 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 Aggregation 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 Aggregation Ports, the Aggregation Port with the numerically lower value of Port Aggregation Priority has the higher priority.
c) Aggregation 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 Aggregation Port of the System with the higher priority, and working downward through the ordered list of Port Aggregation Priority values for the N Aggregation 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 Aggregation 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 LAG (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.

The use of the standby capability is discussed in Annex C.

6.7.2 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 Aggregation 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 Aggregation 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 Aggregation Ports in the aggregation. The original operational Key value should be maintained for the highest priority Aggregation 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 an Aggregation Port can dynamically change its operating characteristics (e.g., data rate, point-to-point operation). It is permissible (and appropriate) for the operational Key value associated with such an Aggregation 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.

6.7.3 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 Aggregation Ports.

6.7.4 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 6.4.14.2.
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 Aggregation Ports supported.

6.7.4.1 Reduced reconfiguration

By removing the constraint that the Aggregator chosen is always the lowest numbered Aggregator associated with the set of Aggregation 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 Aggregation Ports with a given operational Key value, any Aggregation 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 6.4.14.2.

6.7.4.2 Limited Aggregator availability

By removing the constraint that there are always as many Aggregators as Aggregation Ports, an implementation can limit the number of Aggregator Client interfaces available to higher layers while maintaining the ability for each Aggregator to serve multiple Aggregation Ports. This has the same effect as removing the assumption that Aggregators and their associated Aggregation 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 Aggregation Ports.

In this scenario, any Aggregation 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.

7. Management

7.1 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 802.3-2008, 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.

7.1.1 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.

7.1.2 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.

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 preceding items are defined in 7.3 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.

7.2 Managed objects

7.2.1 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 wraparound counters. Wraparound 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, wraparound 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 (7–1).

$$\text{Maximum increment rate specified} \times \left(\frac{\text{higher speed of operation in Mb/s}}{\text{specified speed of operation in Mb/s}} \right) \quad (7\text{–}1)$$

unless otherwise indicated.

7.2.2 Overview of managed objects

Managed objects provide a means to

— Identify a resource
— Control a resource
— Monitor a resource

7.2.2.1 Text description of managed objects

In case of conflict, the formal behaviour definitions in 7.3 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 contained within oDistributedRelay. Since oDistributedRelay is optional, oAggregator can be the top-most managed object class of the Link Aggregation containment tree. The oAggregator managed object class provides the management controls necessary to allow an instance of an Aggregator to be managed.

oDistributedRelay

oDistributedRelay is optional, and when present, is the top-most managed object class of the tree shown in Figure 7-1. Note that this managed object class (or oAggregator, if oDistributedRelay is not present) may be contained within another superior managed object class. Such containment is expected, but is outside the scope of this international standard. The oDistributedRelay managed object class provides the management controls necessary to allow an instance of a Distributed Relay to be managed.

oDistributedRelayIPP

oDistributedRelayIPP is contained within oDistributedRelay. An instance of this managed object class is present for each IPP on the DR Function of a Portal System represented by the oDistributedRelay instance. This managed object class provides the basic management controls necessary to allow an instance of an IPP for the purposes of DRNI.

oIPPDebugInformation

A single instance of oIPPDebugInformation may be contained within oDistributedRelayIPP. This managed object class provides optional additional information that can assist with debugging and fault finding in Systems that support DRNI.

oIPPStats

A single instance of oIPPStats may be contained within oDistributedRelayIPP. This managed object class provides optional additional statistics related to DRCP protocol activity on an instance of an IPP that is involved in DRNI.

7.2.3 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 7-1. 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.

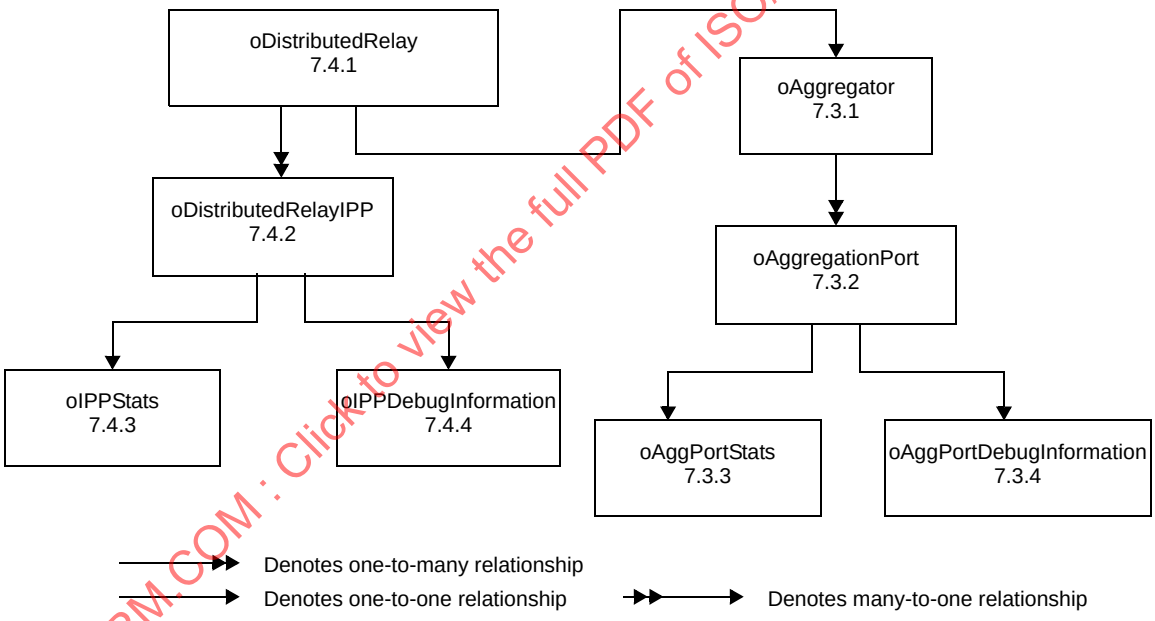


Figure 7-1—Link aggregation entity relationship diagram

7.2.4 Naming

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

7.2.5 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 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, unless stated otherwise. The capabilities and packages for IEEE 802.1AX Management are specified in Table 7–1.

Table 7–1—Link Aggregation capabilities

	Object name	Object type	Operations supported	DTE									
				Basic package (mandatory)	Mandatory package (mandatory)	Recommended package (optional)	Optional package (optional)	Aggregation Port statistics (optional)	Aggregation Port debug information (optional)	Per-Service Frame Distribution package (optional)	DRNI package (optional)	IPP statistics (optional)	IPP debug information (optional)
oAggregator (7.3.1)													
	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								

Table 7–1—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)	Per-Service Frame Distribution package (optional)	DRNI package (optional)	IPP statistics (optional)	IPP debug information (optional)
	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								
	nAggLinkUpNotification	NOTIFICATION			X								
	nAggLinkDownNotification	NOTIFICATION			X								
	aAggPortList	ATTRIBUTE	GET			X							
	aAggCollectorMaxDelay	ATTRIBUTE	GET-SET		X								
	aAggPortAlgorithm	ATTRIBUTE	GET-SET							X			
	aAggPartnerAdminPortAlgorithm	ATTRIBUTE	GET-SET							X			
	aAggConversationAdminLink[]	ATTRIBUTE	GET-SET							X			
	aAggPartnerAdminPortConversationListDigest	ATTRIBUTE	GET-SET							X			
	aAggAdminDiscardWrongConversation	ATTRIBUTE	GET-SET							X			
	aAggAdminServiceConversationMap[]	ATTRIBUTE	GET-SET							X			
	aAggPartnerAdminConvServiceMappingDigest	ATTRIBUTE	GET-SET							X			
oAggregationPort (7.3.2)													
	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								

Table 7–1—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)	Per-Service Frame Distribution package (optional)	DRNI package (optional)	IPP statistics (optional)	IPP debug information (optional)
	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								
	aAggPortOperConversationPasses	ATTRIBUTE	GET							X			
	aAggPortOperConversationCollected	ATTRIBUTE	GET							X			
	aAggPortLinkNumberID	ATTRIBUTE	GET-SET							X			
	aAggPortPartnerAdminLinkNumberID	ATTRIBUTE	GET-SET							X			
	aAggPortWTRTime	ATTRIBUTE	GET-SET							X			
	aAggPortProtocolDA	ATTRIBUTES	GET-SET		X								
oAggPortStats (7.3.3)													
	aAggPortStatsID	ATTRIBUTE	GET					X					
	aAggPortStatsLACPDUsRx	ATTRIBUTE	GET					X					
	aAggPortStatsMarkerPDUsRx	ATTRIBUTE	GET					X					
	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 (7.3.4)													
	aAggPortDebugInformationID	ATTRIBUTE	GET						X				
	aAggPortDebugRxState	ATTRIBUTE	GET						X				
	aAggPortDebugLastRxTime	ATTRIBUTE	GET						X				

Table 7–1—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)	Per-Service Frame Distribution package (optional)	DRNI package (optional)	IPP statistics (optional)	IPP debug information (optional)
	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				
	aAggPortDebugActorCDSChurnState	ATTRIBUTE	GET						X				
	aAggPortDebugPartnerCDSChurnState	ATTRIBUTE	GET						X				
	aAggPortDebugActorCDSChurnCount	ATTRIBUTE	GET						X				
	aAggPortDebugPartnerCDSChurnCount	ATTRIBUTE	GET						X				
oDistributedRelay (7.4.1)													
	aDrniID	ATTRIBUTE	GET								X		
	aDrniDescription	ATTRIBUTE	GET								X		
	aDrniName	ATTRIBUTE	GET-SET								X		
	aDrniPortalAddr	ATTRIBUTE	GET-SET								X		
	aDrniPortalPriority	ATTRIBUTE	GET-SET								X		
	aDrniThreePortalSystem	ATTRIBUTE	GET-SET								X		
	aDrniPortalSystemNumber	ATTRIBUTE	GET-SET								X		
	aDrniIntraPortalLinkList	ATTRIBUTE	GET-SET								X		
	aDrniAggregator	ATTRIBUTE	GET-SET								X		
	aDrniConvAdminGateway[]	ATTRIBUTE	GET-SET								X		
	aDrniNeighborAdminConvGatewayListDigest	ATTRIBUTE	GET-SET								X		
	aDrniNeighborAdminConvPortListDigest	ATTRIBUTE	GET-SET								X		
	aDrniGatewayAlgorithm	ATTRIBUTE	GET-SET								X		
	aDrniNeighborAdminGatewayAlgorithm	ATTRIBUTE	GET-SET								X		
	aDrniNeighborAdminPortAlgorithm	ATTRIBUTE	GET-SET								X		
	aDrniNeighborAdminDRCPState	ATTRIBUTE	GET-SET								X		
	aDrniEncapsulationMethod	ATTRIBUTE	GET-SET								X		
	aDrniIPLEncapMap	ATTRIBUTE	GET-SET								X		

Table 7–1—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)	Per-Service Frame Distribution package (optional)	DRNI package (optional)	IPP statistics (optional)	IPP debug information (optional)
	aDrniNetEncapMap	ATTRIBUTE	GET-SET								X		
	aDrniDRPortConversationPasses	ATTRIBUTE	GET								X		
	aDrniDRGatewayConversationPasses	ATTRIBUTE	GET								X		
	aDrniPSI	ATTRIBUTE	GET								X		
	aDrniPortConversationControl	ATTRIBUTES	GET-SET								X		
	aDrniIntraPortalPortProtocolDA	ATTRIBUTES	GET-SET								X		
oDistributedRelayIPP (7.4.2)													
	aIPPID	ATTRIBUTE	GET								X		
	aIPPPortConversationPasses	ATTRIBUTE	GET								X		
	aIPPGatewayConversationDirection	ATTRIBUTE	GET								X		
	aIPPAdminState	ATTRIBUTE	GET-SET								X		
	aIPPOperState	ATTRIBUTE	GET								X		
	aIPPTimeOfLastOperChange	ATTRIBUTE	GET								X		
oIPPStats (7.4.3)													
	aIPPStatsID	ATTRIBUTE	GET									X	
	aIPPStatsDRCPDUsRx	ATTRIBUTE	GET									X	
	aIPPStatsIllegalRx	ATTRIBUTE	GET									X	
	aIPPStatsDRCPDUsTx	ATTRIBUTE	GET									X	
oIPPDebugInformation (7.4.4)													
	aIPPDebugInformationID	ATTRIBUTE	GET										X
	aIPPDebugDRCPRxState	ATTRIBUTE	GET										X
	aIPPDebugLastRxTime	ATTRIBUTE	GET										X
	aIPPDebugDifferPortalReason	ATTRIBUTE	GET										X
Common Attributes Template													
	aCMCounter	ATTRIBUTE	GET		X	X	X	X	X			X	X

7.3 Management for Link Aggregation

7.3.1 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 MAC. The counts of frames received and transmitted are formulated to reflect the counts that would be expected by the Aggregator 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 Aggregator Client and the Aggregator. However, as LACP and the Marker protocol are, as far as the individual MACs are concerned, part of their Aggregator Client, the RX/TX counters for the individual MACs will reflect both control and data traffic. As counts of errors at the Aggregation 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 Aggregation 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 modeling 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.

7.3.1.1 Aggregator attributes

7.3.1.1.1 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 SMIv2 MIB as an ifIndex—see D.4.1.

7.3.1.1.2 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.

7.3.1.1.3 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.

7.3.1.1.4 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 6, 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 Aggregation 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 Aggregation 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 6.7).

7.3.1.1.5 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.

7.3.1.1.6 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").

7.3.1.1.7 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 6.7.2. This is a 16-bit read-write value. The meaning of particular Key values is of local significance. For an Aggregator that is associated with a Portal, the aAggActorAdminKey has to be different for each Portal System. Specifically the two most significant bits are set to aDrniPortalSystemNumber (7.4.1.1.7). The lower 14 bits may be any value, have to be the same in each Portal System within the same Portal, and have a default of zero.

7.3.1.1.8 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 6.7.2. This is a 16-bit read-only value. The meaning of particular Key values is of local significance.

7.3.1.1.9 aAggMACAddress

ATTRIBUTE

APPROPRIATE SYNTAX:

MACAddress

BEHAVIOUR DEFINED AS:

A 6-octet read-only value carrying the individual MAC address assigned to the Aggregator.

7.3.1.1.10 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.

7.3.1.1.11 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.

7.3.1.1.12 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.

7.3.1.1.13 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.

7.3.1.1.14 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. An operational state of "up" indicates that the Aggregator is available for use by the Aggregator Client; a value of "down" indicates that the Aggregator is not available for use by the Aggregator Client.

7.3.1.1.15 aAggTimeOfLastOperChange

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The time at which the interface entered its current operational state, in terms of centiseconds since the system was last reset. 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. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aAggTimeOfLastOperChange. This value is read-only.

NOTE—aAggTimeOfLastOperChange was defined in terms of the aTimeSinceSystemReset variable of IEEE Std 802.3-2008, F.2.1, in earlier versions of this standard. aTimeSinceSystemReset and ifLastChange have the same meaning.

7.3.1.1.16 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 the sum of the data rate of each link in the aggregation. This attribute is read-only.

7.3.1.1.17 aAggOctetsTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

7.3.1.1.18 aAggOctetsRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4), or frames discarded by the Frame Collection function of the Aggregator (7.3.1.1.26). This value is read-only.

7.3.1.1.19 aAggFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

7.3.1.1.20 aAggFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4), or frames discarded by the Frame Collection function of the Aggregator (7.3.1.1.26). This value is read-only.

7.3.1.1.21 aAggMulticastFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 DA other than the broadcast address. The count does not include frames transmitted by the Aggregator that carry LACP or Marker PDUs (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

7.3.1.1.22 aAggMulticastFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4), or frames discarded by the Frame Collection function of the Aggregator (7.3.1.1.26). This value is read-only.

7.3.1.1.23 aAggBroadcastFramesTxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded by the Frame Distribution function of the Aggregator (7.3.1.1.25). This value is read-only.

7.3.1.1.24 aAggBroadcastFramesRxOK

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4), illegal or unknown protocol frames (7.3.3.1.5, 7.3.3.1.6), or frames discarded by the Frame Collection function of the Aggregator (7.3.1.1.26). This value is read-only.

7.3.1.1.25 aAggFramesDiscardedOnTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 Frame Distribution function of the Aggregator when conversations are reallocated to different Aggregation Ports, due to the requirement to ensure that the conversations are flushed on the old Aggregation Ports in order to maintain proper frame ordering (B.3), or discarded as a result of excessive collisions by Aggregation Ports that are (or have been) members of the aggregation. This value is read-only.

7.3.1.1.26 aAggFramesDiscardedOnRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 Aggregation Ports that are (or have been) members of the aggregation, that were discarded by the Frame Collection function of the Aggregator as they were received on Aggregation Ports whose Frame Collection function was disabled. This value is read-only.

7.3.1.1.27 aAggFramesWithTxErrors

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 Aggregation 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.

7.3.1.1.28 aAggFramesWithRxErrors

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 Aggregation Ports that are (or have been) members of the aggregation, or that were discarded by the Frame Collection function of the Aggregator, or that were discarded by the Aggregator due to the detection of an illegal Slow Protocols PDU (7.3.3.1.6). This value is read-only.

7.3.1.1.29 aAggUnknownProtocolFrames

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 Aggregation Ports that are (or have been) members of the aggregation, due to the detection of an unknown Slow Protocols PDU (7.3.3.1.5). This value is read-only.

7.3.1.1.30 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 (6.3.9, 6.3.14, 6.4.15). An empty list indicates that there are no Aggregation Ports attached. Each integer value in the list carries an aAggPortID attribute value (7.3.2.1.1).

7.3.1.1.31 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.

7.3.1.1.32 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 Aggregator Client or discarding the frame (see 6.2.3.1.1).

7.3.1.1.33 aAggPortAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID and one following octet.

BEHAVIOR DEFINED AS

This object identifies the algorithm used by the Aggregator to assign frames to a Port Conversation ID. Table 6-4 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings.

7.3.1.1.34 aAggPartnerAdminPortAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID and one following octet.

BEHAVIOR DEFINED AS

This object identifies the value for the algorithm of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Table 6-4 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings. Its default value is set to NULL.

7.3.1.1.35 aAggConversationAdminLink[]

ATTRIBUTE

APPROPRIATE SYNTAX

An array of SEQUENCE OF INTEGERs that match the syntax of a Link Number ID ().

BEHAVIOR DEFINED AS

There are 4096 aAggConversationAdminLink[] variables, aAggConversationAdminLink[0] through aAggConversationAdminLink[4095], indexed by Port Conversation ID. Each contains administrative values of the link selection priority list for the referenced Port Conversation ID. This selection priority list is a sequence of Link Number IDs for each Port Conversation ID, in the order of preference, highest to lowest, for the corresponding link to carry that Port Conversation ID. A 16-bit zero value is used to indicate that no link is assigned to carry the associated Port Conversation ID.

NOTE 1—This mapping of Port Conversation IDs to Link Number IDs is the fundamental administrative input. An equivalent mapping of Port Conversation IDs to Port IDs [Conversation_PortList[] ()] is derived from this and used internally.

NOTE 2—When a network administrator issues a command for selection rules, provided by aAggConversationAdminLink[], and accompanied with a non-zero value for aAggPortWTRTime (7.3.2.1.29) for all associated Aggregation Ports, the ChangeActorOperDist is set as specified in 6.6.2.2. A value of 100 for the aAggPortWTRTime indicates a non-revertive mode of operation, and the WTR_timer will be kept to the value 100.

7.3.1.1.36 aAggPartnerAdminPortConversationListDigest

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS

The value for the digest of the prioritized Port Conversation ID-to-Link Number ID assignments of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Its default value is set to NULL.

7.3.1.1.37 aAggAdminDiscardWrongConversation

ATTRIBUTE

APPROPRIATE SYNTAX

BOOLEAN

BEHAVIOR DEFINED AS

The administrative value that determines what the Aggregator does with a frame that is received from an Aggregation Port with a Port Conversation ID that is not included in the Collection_Conversation_Mask. The value "TRUE" indicates that such frames are to be discarded, and the value "FALSE" that they are to be forwarded. This variable needs to be set to "TRUE," if bidirectional congruity (8.2.1) is required. Its value is set to "TRUE" by default.

7.3.1.1.38 aAggAdminServiceConversationMap[]

ATTRIBUTE

APPROPRIATE SYNTAX

An array of SEQUENCE OF INTEGERs, that match the syntax of Service IDs (8.2.2).

BEHAVIOR DEFINED AS

There are 4096 aAggAdminServiceConversationMap[] variables, aAggAdminServiceConversationMap[0] through aAggAdminServiceConversationMap[4095], indexed by Port Conversation ID. Each contains, in general, a set of Service IDs (8.2.2), unique within the array. If the Service IDs are representing VIDs, only a single VID is used, while in the case that Service IDs are representing I-SIDs, more than one I-SIDs are possible. Service IDs not contained in the map are not mapped to any Port Conversation ID and will be discarded.

7.3.1.1.39 aAggPartnerAdminConvServiceMappingDigest

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS

The value for the digest of the Port Conversation ID-to-Service ID assignments of the Partner System, assigned by administrator or System policy for use when the Partner's information is unknown. Its default value is set to NULL.

7.3.1.2 Aggregator Notifications

7.3.1.2.1 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.

7.3.1.2.2 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.

7.3.2 Aggregation Port managed object class

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

7.3.2.1 Aggregation Port Attributes

7.3.2.1.1 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 SMIv2 MIB as an ifIndex—see D.4.1.

7.3.2.1.2 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.

7.3.2.1.3 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.

7.3.2.1.4 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.

7.3.2.1.5 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.

7.3.2.1.6 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.

7.3.2.1.7 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.

7.3.2.1.8 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.

7.3.2.1.9 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.

7.3.2.1.10 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.

7.3.2.1.11 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.

7.3.2.1.12 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.

7.3.2.1.13 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.

7.3.2.1.14 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.

7.3.2.1.15 aAggPortActorPortPriority

ATTRIBUTE

APPROPRIATE SYNTAX:
INTEGER

BEHAVIOUR DEFINED AS:
The priority value assigned to this Aggregation Port. This 16-bit value is read-write.

NOTE—In the case of DRNI (Clause 9), the two least significant bits of the priority for each Aggregation Port in a Distributed Relay's Aggregator Port will be ignored because these bits are used to encode the Portal System Number [item e) in 9.3.4].

7.3.2.1.16 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.

7.3.2.1.17 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.

7.3.2.1.18 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.

7.3.2.1.19 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.

7.3.2.1.20 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 (6.4.2) 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.

7.3.2.1.21 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 (6.4.2) as transmitted by the Actor in LACPDUs. The bit allocations are as defined in 7.3.2.1.20. This attribute value is read-only.

7.3.2.1.22 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 7.3.2.1.20. This attribute value is read-write. The assigned value is used in order to achieve manually configured aggregation.

7.3.2.1.23 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 7.3.2.1.20. In the absence of an active protocol Partner, this value may reflect the manually configured value aAggPortPartnerAdminState. This attribute value is read-only.

7.3.2.1.24 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”).

7.3.2.1.25 aAggPortOperConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS

A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that the Port Conversation ID is distributed through this Aggregation Port, and a 0 indicates that it cannot. aAggPortOperConversationPasses is referencing the current value of Port_Oper_Conversation_Mask (6.6.2.2).

7.3.2.1.26 aAggPortOperConversationCollected

ATTRIBUTE

APPROPRIATE SYNTAX

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS

A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that the Port Conversation ID is collected through this Aggregation Port, and a 0 indicates that it cannot. aAggPortOperConversationPasses is referencing the current value of Collection_Conversation_Mask (6.6.1.1.2).

7.3.2.1.27 aAggPortLinkNumberID

ATTRIBUTE

APPROPRIATE SYNTAX

INTEGER, 0 to 65535

BEHAVIOUR DEFINED AS:

The Link Number ID value configured for this Aggregation Port by the System's administrator. When the Link Number ID value matches one of the non-zero values in the selection prioritized lists in aAggConversationAdminLink[] (7.3.1.1.35), then this Aggregation Port must be configured to have an aAggPortActorAdminKey value that matches the aAggActorAdminKey of the Aggregator used by the LAG of the links specified in aAggConversationAdminLink[]. Its default value is set to aAggPortActorPort (7.3.2.1.14).

NOTE—In the case of DRNI, the match of the aAggActorAdminKey to aAggPortActorAdminKey values excludes the first two bits identifying the individual Portal System in the Portal. If the network administrator fails to configure the proper values for the aAggPortActorAdminKey variables in all of the Aggregators Ports attached to a Portal, the DRCP (9.4) and the variable Port_Oper_Conversation_Mask (6.6.2.2) prevent looping and/or duplicate delivery, if necessary, by discarding frames belonging to misconfigured Conversations.

7.3.2.1.28 aAggPortPartnerAdminLinkNumberID

ATTRIBUTE

APPROPRIATE SYNTAX

INTEGER, 0 to 65535

BEHAVIOUR DEFINED AS:

The value for the Link Number ID of the Partner System for this Aggregation Port, assigned by administrator or System policy for use when the Partner's information is unknown. Its default value is set to 0.

7.3.2.1.29 aAggPortWTRTime

ATTRIBUTE

APPROPRIATE SYNTAX

INTEGER

BEHAVIOUR DEFINED AS:

The wait-to-restore (WTR) period accompanying selection rules set by aAggConversationAdminLink[] in a command issued by a network administrator. It may be configured in steps of 1 min between 5 min and 12 min, while two additional special values are also used. The value 0 indicates revertive and is the default value. The value 100 indicates non-revertive mode of operation, and the WTR_timer will be kept to the value 100.

7.3.2.2 Aggregation Port Extension Attributes

7.3.2.2.1 aAggPortProtocolDA

ATTRIBUTE

APPROPRIATE SYNTAX

MACAddress

BEHAVIOR DEFINED AS

A 6-octet read-write MACAddress value specifying the DA to be used when sending Link Aggregation Control and Marker PDUs on this Aggregation Port, corresponding to the value of Protocol_DA in 6.2.8.1.2, 6.2.10.1.3, and 6.5.4.2.1. The default value shall be the IEEE 802.3 Slow_Protocols_Multicast address.

7.3.3 Aggregation Port Statistics managed object class

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

7.3.3.1 Aggregation Port Statistics attributes

7.3.3.1.1 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.

7.3.3.1.2 aAggPortStatsLACPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.3.1.3 aAggPortStatsMarkerPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.3.1.4 aAggPortStatsMarkerResponsePDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.3.1.5 aAggPortStatsUnknownRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 802.3-2008, Annex 57A.4), but contain an unknown PDU, or

— Are addressed to the Slow Protocols group MAC Address (IEEE Std 802.3-2008, Annex 57A.3), but do not carry the Slow Protocols Ethernet Type. This value is read-only.

7.3.3.1.6 aAggPortStatsIllegalRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 802.3-2008, Annex 57A.4), but contain a badly formed PDU or an illegal value of Protocol Subtype (IEEE Std 802.3-2008, Annex 57A.3). This value is read-only.

7.3.3.1.7 aAggPortStatsLACPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.3.1.8 aAggPortStatsMarkerPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.3.1.9 aAggPortStatsMarkerResponsePDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.4 Aggregation Port Debug Information managed object class

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

7.3.4.1 Aggregation Port Debug Information attributes

7.3.4.1.1 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.

7.3.4.1.2 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.

7.3.4.1.3 aAggPortDebugLastRxTime

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The time at which the last LACPDU was received by this Aggregation Port, in terms of centiseconds since the system was last reset. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aAggPortDebugLastRxTime. This value is read-only.

NOTE—aAggPortDebugLastRxTime was defined in terms of the aTimeSinceSystemReset variable of IEEE Std 802.3-2008, Annex F, F.2.1, in earlier versions of this standard. aTimeSinceSystemReset and ifLastChange have the same meaning.

7.3.4.1.4 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 (6.4.15) 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.

7.3.4.1.5 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.

7.3.4.1.6 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 (6.4.17) 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.

7.3.4.1.7 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 (6.4.17) 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.

7.3.4.1.8 aAggPortDebugActorChurnCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.4.1.9 aAggPortDebugPartnerChurnCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.4.1.10 aAggPortDebugActorSyncTransitionCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (6.4.15) has entered the IN_SYNC state. This value is read-only.

7.3.4.1.11 aAggPortDebugPartnerSyncTransitionCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (6.4.15) has entered the IN_SYNC state. This value is read-only.

7.3.4.1.12 aAggPortDebugActorChangeCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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.

7.3.4.1.13 aAggPortDebugPartnerChangeCount

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized 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 (6.3.6.1) for this Aggregation Port has changed. This value is read-only.

7.3.4.1.14 aAggPortDebugActorCDSChurnState

ATTRIBUTE

APPROPRIATE SYNTAX:

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

churn

BEHAVIOUR DEFINED AS:

This managed object is applicable only when Conversation-sensitive frame collection and distribution as specified in 6.6 is supported. The state of the Actor CDS Churn Detection machine (6.6.2.7) for the Aggregation Port. A value of “noChurn” indicates that the state machine is in either the NO_ACTOR_CDS_CHURN or the ACTOR_CHURN_CDS_MONITOR state, and “churn” indicates that the state machine is in the ACTOR_CDS_CHURN state. This value is read-only.

7.3.4.1.15 aAggPortDebugPartnerCDSChurnState

ATTRIBUTE

APPROPRIATE SYNTAX:

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

BEHAVIOUR DEFINED AS:

This managed object is applicable only when Conversation-sensitive frame collection and distribution as specified in 6.6 is supported. The state of the Partner CDS Churn Detection machine (6.6.2.7) for the Aggregation Port. A value of “noChurn” indicates that the state machine is in either the NO_PARTNER_CDS_CHURN or the PARTNER_CDS_CHURN_MONITOR state, and “churn” indicates that the state machine is in the PARTNER_CDSCHURN state. This value is read-only.

7.3.4.1.16 aAggPortDebugActorCDSChurnCount

ATTRIBUTE

APPROPRIATE SYNTAX:

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

BEHAVIOUR DEFINED AS:

This managed object is applicable only when Conversation-sensitive frame collection and distribution as specified in 6.6 is supported. Count of the number of times the Actor CDS Churn state machine has entered the ACTOR_CDS_CHURN state. This value is read-only.

7.3.4.1.17 aAggPortDebugPartnerCDSChurnCount

ATTRIBUTE

APPROPRIATE SYNTAX:

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

BEHAVIOUR DEFINED AS:

This managed object is applicable only when Conversation-sensitive frame collection and distribution as specified in 6.6 is supported. Count of the number of times the Partner CDS Churn state machine has entered the PARTNER_CDS_CHURN state. This value is read-only.

7.4 Management for Distributed Resilient Network Interconnect

7.4.1 Distributed Relay Managed Object Class

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

7.4.1.1 Distributed Relay Attributes

7.4.1.1.1 aDrniID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

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

NOTE—The aDrniID is represented in the SMIv2 MIB as an ifIndex—see D.5.

7.4.1.1.2 aDrniDescription

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOUR DEFINED AS:

A human-readable text string containing information about the Distribute Relay. This string is read-only. The contents are vendor specific.

7.4.1.1.3 aDrniName

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOUR DEFINED AS:

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

7.4.1.1.4 aDrniPortalAddr

ATTRIBUTE

APPROPRIATE SYNTAX:

A SEQUENCE OF 6 OCTETS that match the syntax of a 48-bit MAC Address

BEHAVIOUR DEFINED AS:

A read-write identifier of a particular Portal. aDrniPortalAddr has to be unique among at least all of the potential Portal Systems to which a given Portal System might be attached via an IPL Intra-Portal Link. Also used as the Actor's System ID (6.3.2) for the emulated system.

7.4.1.1.5 aDrniPortalPriority

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

A 2-octet read-write value indicating the priority value associated with the Portal's System ID. Also used as the Actor's System Priority (6.3.2) for the emulated system.

7.4.1.1.6 aDrniThreePortalSystem

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOUR DEFINED AS:

A read-write Boolean value indicating whether this Portal System is part of a Portal consisting of three Portal Systems or not. Value 1 stands for a Portal of three Portal Systems, value 0 stands for a Portal of two or one Portal Systems. The default value is 0.

7.4.1.1.7 aDrniPortalSystemNumber

ATTRIBUTE

APPROPRIATE SYNTAX

A Portal System Number, which is an integer in the range 1 through 3 inclusive.

BEHAVIOR DEFINED AS

A read-write identifier of this particular Portal System within a Portal. It is the responsibility of the network administrator to ensure that these numbers are unique among the Portal Systems with the same aDrniPortalAddr (7.4.1.1.4).

7.4.1.1.8 aDrniIntraPortalLinkList

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF INTEGERs that match the syntax of an Interface Identifier.

BEHAVIOR DEFINED AS

Read-write list of the Interface Identifiers of the Ports to the Intra-Portal Links assigned to this Distributed Relay. Each Interface Identifier, a Port ID (6.3.4), has the two least significant bits of its Port Priority (7.3.2.1.15) configured to match the Portal System Number of the attached Portal System. The number of IPLs in the list depends on the Portal topology. For a Portal of three Portal Systems two or three IPLs can be used, for a Portal of two Portal Systems a single IPL is required and for a single Portal System no IPL is required.

7.4.1.1.9 aDrniAggregator

ATTRIBUTE

APPROPRIATE SYNTAX

An INTEGER that matches the syntax of an Interface Identifier.

BEHAVIOR DEFINED AS

Read-write Interface Identifier of the Aggregator Port assigned to this Distributed Relay.

7.4.1.1.10 aDrniConvAdminGateway[]

ATTRIBUTE

APPROPRIATE SYNTAX

An array of SEQUENCE OF INTEGERs that match the syntax of Portal System Number.

BEHAVIOR DEFINED AS

There are 4096 aDrniConvAdminGateway[] variables, aDrniConvAdminGateway[0] through aDrniConvAdminGateway[4095], indexed by Gateway Conversation ID. Each contains administrative values of the Gateway selection priority list for the Distributed Relay for the referenced Gateway Conversation ID. This selection priority list, a sequence of

integers for each Gateway Conversation ID, is a list of Portal System Numbers in the order of preference, highest to lowest, for the corresponding preferred Portal System's Gateway to carry that Conversation.

NOTE—To the extent that the network administrator fails to configure the same values for the aDrniConvAdminGateway[] variables in all of the DR Functions of a Portal, frames can be misdirected. The DRCP (9.4) detects such misconfiguration.

7.4.1.1.11 aDrniNeighborAdminConvGatewayListDigest

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS

The value for the digest of the prioritized Gateway Conversation ID-to-Gateway assignments of the Neighbor Portal System, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Its default value is set to NULL.

7.4.1.1.12 aDrniNeighborAdminConvPortListDigest

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 16-octet MD5 Digest.

BEHAVIOR DEFINED AS

The value for the digest of the prioritized Port Conversation ID-to-Aggregation Port assignments of the Neighbor Portal System, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Its default value is set to NULL.

7.4.1.1.13 aDrniGatewayAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of an OUI or CID and one following octet.

BEHAVIOR DEFINED AS

This object identifies the algorithm used by the DR Function to assign frames to a Gateway Conversation ID. Table 9-7 provides the IEEE 802.1 OUI (00-80-C2) Gateway Algorithm encodings.

7.4.1.1.14 aDrniNeighborAdminGatewayAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID and one following octet.

BEHAVIOR DEFINED AS

This object identifies the value for the Gateway algorithm of the Neighbor Portal System, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Table 9-7 provides the IEEE 802.1 OUI (00-80-C2) Gateway Algorithm encodings. Its default value is set to NULL.

7.4.1.1.15 aDrniNeighborAdminPortAlgorithm

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of a 3-octet OUI or CID and one following octet.

BEHAVIOR DEFINED AS

This object identifies the value for the Port Algorithm of the Neighbor Portal System, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Table 6-4 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings. Its default value is set to NULL.

7.4.1.1.16 aDrniNeighborAdminDRCPState

ATTRIBUTE

APPROPRIATE SYNTAX:

BIT STRING [SIZE (1..8)]

BEHAVIOUR DEFINED AS:

A string of 8 bits, corresponding to the administrative values of DRCP_State [item s) in 9.4.3.2] as transmitted by this Portal System in DRCPDUs. The first bit corresponds to bit 0 of DRCP_State (Home_Gateway), the second bit corresponds to bit 1 (Neighbor_Gateway), the third bit corresponds to bit 2 (Other_Gateway), the fourth bit corresponds to bit 3 (IPP_Activity), the fifth bit corresponds to bit 4 (DRCP_Timeout), the sixth bit corresponds to bit 5 (Gateway_Sync), the seventh bit corresponds to bit 6 (Port_Sync), and the eighth bit corresponds to bit 7 (Expired). These values allow administrative control over the values of Home_Gateway, Neighbor_Gateway, Other_Gateway, IPP_Activity, and DRCP_Timeout. Their values are by default set to FALSE. This attribute value is read-write.

7.4.1.1.17 aDrniEncapsulationMethod

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF OCTETS consisting of an OUI or CID and one following octet.

BEHAVIOR DEFINED AS

This managed object is applicable only when Network / IPL sharing by time (9.3.2.1) or Network / IPL sharing by tag (9.3.2.2) or Network / IPL sharing by encapsulation (9.3.2.3) is supported. The object identifies the value representing the encapsulation method that is used to transport IPL frames to the Neighbor Portal System when the IPL and network link are sharing the same physical link. It consists of the 3-octet OUI or CID identifying the organization that is responsible for this encapsulation and one following octet used to identify the encapsulation method defined by that organization. Table 9-11 provides the IEEE 802.1 OUI (00-80-C2) encapsulation method encodings. A Default value of 0x00-80-C2-00 indicates that the IPL is using a separate physical or Aggregation link. A value of 1 indicates that Network / IPL sharing by time (9.3.2.1) is used. A value of 2 indicates that the encapsulation method used is the same as the one used by network frames and that Network / IPL sharing by tag (9.3.2.2) is used.

7.4.1.1.18 aDrniIPLEncapMap

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF INTEGERs, indexed by Gateway Conversation ID.

BEHAVIOR DEFINED AS

This managed object is applicable only when Network / IPL sharing by tag (9.3.2.2) or Network / IPL sharing by encapsulation (9.3.2.3) is supported. Each entry represents the value of the identifier used for an IPL frame associated with that Gateway Conversation ID for the encapsulation method specified in 7.4.1.1.17.

7.4.1.1.19 aDrniNetEncapMap

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF INTEGERs, indexed by Gateway Conversation ID.

BEHAVIOR DEFINED AS

This managed object is applicable only when Network / IPL sharing by tag (9.3.2.2) is supported. Each entry represents the translated value of the identifier used for a network frame associated with that Gateway Conversation ID when the method specified in 7.4.1.1.17 is the Network / IPL sharing by tag method specified in 9.3.2.2 and the network frames need to share the tag space used by IPL frames.

7.4.1.1.20 aDrniDRPortConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS

A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that the Port Conversation ID is allowed to be distributed through this DR Function's Aggregator, and a 0 indicates that it cannot. aDrniDRPortConversationPasses is referencing the current value of Drni_Portal_System_Port_Conversation ().

7.4.1.1.21 aDrniDRGatewayConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS

A read-only current operational vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates that the Gateway Conversation ID is allowed to pass through this DR Function's Gateway, and a 0 indicates that it cannot. aDrniDRGatewayConversationPasses is referencing the current value of Drni_Portal_System_Gateway_Conversation ().

7.4.1.1.22 aDrniPSI

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOUR DEFINED AS:

A read-only Boolean value providing the value of PSI, which indicates whether this Portal System is isolated from the other Portal Systems within the same Portal ("TRUE") or not ("FALSE").

7.4.1.1.23 aDrniPortConversationControl

ATTRIBUTE

APPROPRIATE SYNTAX:

BOOLEAN

BEHAVIOUR DEFINED AS:

A read-write Boolean value that controls the operation of the updateDRFHomeState (9.4.11). When set to "TRUE" the Home Gateway Vector is set equal to Drni_Portal_System_Port_Conversation. Setting this object to "TRUE" is only possible when the Gateway algorithm and the Port algorithm use the same distributions methods. The default is "FALSE," indicating that the Home Gateway Vector is controlled by the network control protocol.

7.4.1.1.24 aDrniIntraPortalPortProtocolDA

ATTRIBUTE

APPROPRIATE SYNTAX

A SEQUENCE OF 6 OCTETS that match the syntax of a 48-bit MAC Address

BEHAVIOR DEFINED AS

A 6-octet read-write MAC Address value specifying the DA to be used when sending DRCPDUs, corresponding to the value of DRCP_Protocol_DA in 9.4.4.1.3. Its values is one of the addresses selected from Table 9-6 and its default shall be the IEEE 802.1 Nearest non-TPMR Bridge group address (01-80-C2-00-00-03).

7.4.2 IPP Managed Objects Class

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

7.4.2.1 IPP Attributes

7.4.2.1.1 aIPPID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

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

NOTE—The aIPPID is represented in the SMIv2 MIB as an ifIndex—see D.5.

7.4.2.1.2 aIPPPortConversationPasses

ATTRIBUTE

APPROPRIATE SYNTAX

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS

A read-only current operational vector of Boolean values, with one value for each possible Port Conversation ID. A 1 indicates that the Port Conversation ID is allowed to be transmitted through this IPP Intra-Portal Port, and a 0 indicates that it cannot.

aIPPPortConversationPasses is referencing the current value of Ipp_Port_Conversation_Passes (9.3.4.3).

7.4.2.1.3 aIPPGatewayConversationDirection

ATTRIBUTE

APPROPRIATE SYNTAX

BIT STRING [SIZE (4096)]

BEHAVIOR DEFINED AS

A read-only current operational vector of Boolean values, with one value for each possible Gateway Conversation ID. A 1 indicates that the Gateway Conversation ID is assigned to Gateways reachable through this IPP Intra-Portal Port, and a 0 indicates that the Gateway for the indexed Gateway Conversation ID is not reachable through this IPP. aIPPGatewayConversationDirection is referencing the current value of Ipp_Gateway_Conversation_Direction (9.3.4.3).

7.4.2.1.4 aIPPAdminState

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 IPP. A value of “up” indicates that the operational state of the IPP (aIPPOperState) is permitted to be either “up” or “down.” A value of “down” forces the operational state of the IPP to be “down”. A GET operation returns the current administrative state. A SET operation changes the administrative state to a new value.

7.4.2.1.5 aIPPOperState

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 IPP. The operational state is “up” if the IPL is operational, and if the value of aIPPAdminState for the IPP is also “up.” If the IPL is not operational, or if the administrative state of the IPP (aIPPAdminState) is “down,” then the operational state is “down.” An operational state of “up” indicates that the IPP is available for use by the DR Function; a value of “down” indicates that the IPP is not available for use by the DR Function.

7.4.2.1.6 aIPPTimeOfLastOperChange

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The time at which the interface entered its current operational state, in terms of centiseconds since the system was last reset. 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. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aIPPTimeOfLastOperChange. This value is read-only.

7.4.3 IPP Statistics managed object class

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

7.4.3.1 IPP Statistics attributes

7.4.3.1.1 aIPPStatsID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

This read-only attribute identifies an IPP 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 oDistributedRelayIPP managed object.

7.4.3.1.2 aIPPStatsDRCPDUsRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has an expected increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of valid DRCPDUs received on this IPP. This value is read-only.

7.4.3.1.3 aIPPStatsIllegalRx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has an expected increment rate of 50 counts per second.

BEHAVIOUR DEFINED AS:

The number of frames received that carry the DRCP Ethernet Type value (9.4.2.4), but contain a badly formed PDU. This value is read-only.

7.4.3.1.4 aIPPStatsDRCPDUsTx

ATTRIBUTE

APPROPRIATE SYNTAX:

Generalized counter. This counter has an expected increment rate of 5 counts per second.

BEHAVIOUR DEFINED AS:

The number of DRCPDUs transmitted on this IPP. This value is read-only.

7.4.4 IPP Debug Information managed object class

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

7.4.4.1 IPP Debug Information attributes

7.4.4.1.1 aIPPDebugInformationID

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

This read-only attribute identifies an DRCP 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 oDistributedRelayIPP managed object.

7.4.4.1.2 aIPPDebugDRCPRxState

ATTRIBUTE

APPROPRIATE SYNTAX:

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

BEHAVIOUR DEFINED AS:

This attribute holds the value “current” if the DRCPDU Receive state machine for the IPP is in the CURRENT state, “expired” if the DRCPDU Receive state machine is in the EXPIRED state, “defaulted” if the DRCPDU Receive state machine is in the DEFAULTED state, “initialize” if the DRCPDU Receive state machine is in the INITIALIZE state, or “reportToManagement” if the Receive state machine is in the REPORT_TO_MANAGEMENT state. This value is read-only.

7.4.4.1.3 aIPPDebugLastRxTime

ATTRIBUTE

APPROPRIATE SYNTAX:

INTEGER

BEHAVIOUR DEFINED AS:

The time at which the last DRCPDU was received by this IPP, in terms of centiseconds since the system was last reset. The ifLastChange object in the Interfaces MIB defined in IETF RFC 2863 is a suitable object for supplying a value for aIPPDebugLastRxTime. This value is read-only.

7.4.4.1.4 aIPPDebugDifferPortalReason

ATTRIBUTE

APPROPRIATE SYNTAX:

A PrintableString, 255 characters max.

BEHAVIOUR DEFINED AS:

A human-readable text string indicating the most recent set of variables that are responsible for setting the variable Differ_Portal or Differ_Conf_Portal (9.4.8) on this IPP to TRUE. This value is read-only.

8. Frame distribution and collection algorithms

One specific algorithm for frame distribution and collection is defined, in 8.2.

8.1 Conversation Identifiers

The number of different conversations, each of which has to be delivered in order, can be very large. Some activities, particularly ensuring frame ordering using Conversation-sensitive frame collection and distribution (6.6), require that functional elements in different Systems maintain and exchange information applying to particular conversations. The enumeration of large numbers of conversations (e.g., conversations identified by a 24-bit I-SID) is impractical for a protocol such as LACP.

Therefore, a Conversation Identifier (or Conversation ID) is defined as a value in the range 0 through 4095. By administrative means, every possible conversation is assigned to a single Conversation ID value for each supported Conversation ID type. More than one conversation can be assigned to a Conversation ID. It is not necessary that every Conversation ID value have any conversations assigned to it. In this standard, several types of Conversation ID are specified for different uses.

8.2 Per-service frame distribution

A System may implement Per-service frame distribution, as defined in this subclause.

8.2.1 Goals and objectives

Distributing frames to physical links by service ID, as defined in 8.2, provides the following:

a) **Connectivity Fault Management congruity**—The only thing that Connectivity Fault Management (CFM, IEEE Std 802.1Q-2014, Clause 18) PDUs have in common with the data that they protect is, in general, the service ID(s) that they share. Per-service frame distribution ensures that the CFM PDUs traverse the same physical links as their data.
b) **Bidirectional congruity**—Providing a means for the two ends of an Aggregation Group to use the same physical link in both directions for a given service ensures that a failed link or a link experiencing an excessive error rate will affect the fewest possible number of services and in general provide support for protocols that have strict symmetry requirements on their transmit and receive paths, e.g., Precision Time Protocol (PTP) in IEEE Std 1588™-2008.
c) **Ingress predictability**—Ingress metering is often applied on a per-service basis. Confining a service to a single physical link localizes the meter, facilitating this process.

NOTE—An Aggregation System or Portal running Per-service frame distribution can interoperate with an Aggregation System or Portal not running Per-service frame distribution. However, only the Link Aggregation goals in 6.1.1 can be met, not the goals listed in this section.

8.2.2 Overview

A device implementing Link Aggregation, may implement Per-service frame distribution. When enabled, Per-service frame distribution determines how Port Conversation IDs are derived from frames by the Aggregator and, in DRNI systems, the Distributed Relay. Whether Per-service frame distribution is enabled on either end of a particular Link Aggregation Group is an administrative choice.

When enabled, Per-service frame distribution distributes frames to physical links according to their Service Identifier (Service ID). The following tag types defined by IEEE Std 802.1Q-2011, Table 9–1, are supported by Per-service frame distribution:

a) Customer VLAN Tag (VID field, 9.6 of IEEE Std 802.1Q-2011);
b) Service VLAN Tag or Backbone VLAN Tag (VID field, 9.6 of IEEE Std 802.1Q-2011); and
c) Backbone Service Instance Tag (I-SID field, 9.7 of IEEE Std 802.1Q-2011).

The active function (Frame Distributor, Frame Collector, or DR Function) examines each frame presented to it. If the first item in the M_UNITDATA.request or M_UNITDATA.indication mac_service_data_unit parameter is the one tag from the preceding list for which active function is configured, encapsulated as appropriate to the medium according to 9.4 of IEEE Std 802.1Q-2011, then the 12-bit VID field or 24-bit I-SID field of the tag is used to determine the Service ID. A value of zero is used if the first item in the mac_service_data_unit is not the appropriate tag.

A System supporting Per-service frame distribution (6.6) shall support Conversation-sensitive frame collection and distribution, and shall support classification by Customer VLAN Tag for C-tagged interfaces and classification by Service VLAN tag for S-tagged and B-tagged interfaces. A System supporting Per-service frame distribution may support service classification by Backbone Service Instance Tag. A Backbone Edge Bridge supporting Per-service frame distribution on its Provider Instance Ports (PIPs, 6.10 of IEEE Std 802.1Q-2014) or Customer Backbone Ports (CBPs, Clause 6.10 of IEEE Std 802.1Q-2011) shall support service classification by Backbone Service Instance Tag on those ports. A System supporting Per-service frame distribution may support classification by any other methods that are identified in aAggPortAlgorithm (7.3.1.1.33).

8.2.3 Port Conversation Identifiers

Per-service frame distribution is defined in terms of Port Conversation IDs (8.1). For 12-bit tags, both VLAN IDs and Port Conversation IDs have 4095 values available, so Port Conversation ID can be synonymous with tag value, and no mapping is needed. However, when 24-bit I-SIDs are used as Service IDs, mapping to Port Conversation IDs is necessary. In this case both Aggregators at the two ends of the LAG have to be configured with the same Service ID to Port Conversation ID mapping entries (7.3.1.1.38). Conversation-sensitive LACPDUs exchange digests of the tables containing these entries (6.4.2.4.4) to verify that the mapping table configuration is coordinated.

9. Distributed Resilient Network Interconnect

This clause describes the elements of Distributed Resilient Network Interconnect (DRNI) as follows:

a) An overview of DRNI is given in 9.1.
b) Distributed Relay is introduced in 9.2.
c) The operation of the Distributed Relay is described in detail in 9.3.
d) Distributed Relay Control Protocol (DRCP) is described in 9.4.

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

9.1 Goals and objectives

As described in Clause 6, Link Aggregation creates a Link Aggregation Group (LAG) that is a collection of one or more physical links that appears, to higher layers, to be a single logical link. The LAG has two ends, each terminating in an Aggregation System.

DRNI expands the concept of Link Aggregation so that, at either one or both ends of a LAG, the single Aggregation System is replaced by a Portal, composed from one to three Portal Systems.

The DRNI, as defined in this clause, provides the following:

a) **Link Aggregation**—DRNI provides benefits of Link Aggregation, as listed in 6.1.1, item a) through item n).
b) **Portals**—Connections between two cooperating sets of Systems (two Portals) are supported, in contrast to Link Aggregation [item o) in 6.1.1], so that connectivity between two networks can be maintained despite the failure of an entire System (and its connected links) belonging to a Portal.
c) **Compatibility**—A multi-System Portal can connect to a single-System Portal or to an Aggregation System compliant with Clause 6 or with previous versions of this Standard.
d) **Administrative isolation**—A DRNI Link Aggregation Group can connect Portals in networks that are under separate administration, running different fault recovery protocols.
e) **Administrative independence**—The specification of DRNI to interconnect separate networks does not introduce new requirements on either networks' existing control protocols.
f) **Inter-network fault isolation**—The failure or recovery of a link or node in one network, requiring a reaction by that network's control protocols, can be hidden by DRNI from the second network's control protocols. Thus, super-networks can be created out of separate networks interconnected via DRNI, without propagating one network's fault and recovery events throughout the super-network.
g) **Network-**DRNI **fault isolation**—The failure or recovery of a link between two Portals can be hidden by DRNI from both networks' control protocols.
h) **Rapid fault recovery**—Means for the Systems in a Portal to communicate are provided so that they can cooperate to respond rapidly to failure or recovery events, typically on the order of milliseconds for link down events and 1 second or less for link up events.
i) **Extended faults**—Optional elements of DRNI can support three Systems in a Portal, so that fault redundancy can be provided even while a System is added or removed from a Portal.
j) **Distribution independence**—The frame distribution algorithm used to satisfy network requirements can be different from the algorithm used to assign frames to the Aggregation Ports of a LAG.

The DRNI, as specified in this clause, does not support the following:

k) **Multipoint Aggregations**—Connections among more than two Portals or Aggregation Systems are not supported.
l) **Virtual System emulation**—DRNI makes no attempt to provide all of the mechanisms required to make two Systems, such as two Bridges or two Routers, appear, to all other Systems, even those in their own network, to be a single (virtual) System. But, neither does DRNI prevent the Systems employing it to emulate a single virtual System.
m) **More than three Systems in a Portal**—In general, a fault-tolerant Portal comprising more than three Systems would require some protocol mechanism, not provided by this standard, to select forwarding paths and to prevent endless forwarding loops.

9.2 Distributed Relay

DRNI is created by using a Distributed Relay to interconnect two or three Systems, each running Link Aggregation, to create a Portal. Each System in the Portal (i.e., each Portal System) runs Link Aggregation with a single Aggregator. The Distributed Relay enables the Portal Systems to jointly terminate a LAG. To all other Systems to which the Portal is connected, the LAG appears to terminate in a separate emulated System created by the Portal Systems.

Figure 9-1 illustrates the starting point for describing the Distributed Relay. The (three) network links depicted in the Figure 9-1 and discussed in this standard correspond to physical or logical links that are visible to and are under the control of network protocols. In this diagram, Systems **A** and **B** each are characterized by performing a "Function 1," which is presumably some kind of packet relay function, e.g., a router or a bridge. "Function 1" could just as well be a file server operation, in which case the outside two "ports" on each System would likely not be present. Each System runs a single instance of a Link Aggregation sublayer. Let us suppose that it is desired that the shaded ports be associated into a Portal with a Distributed Relay.

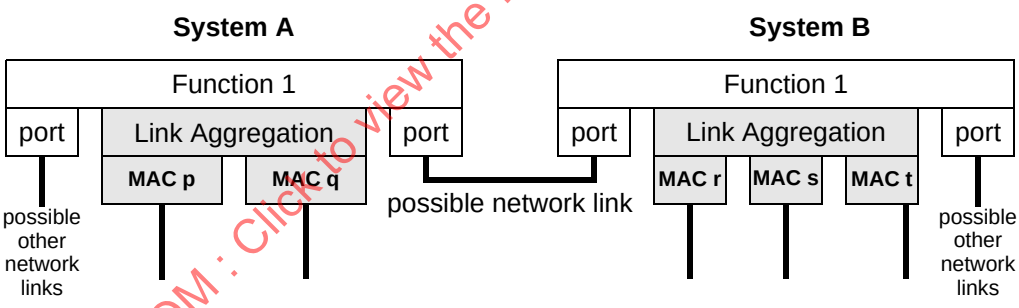


Figure 9-1—Distributed Relay: starting point

Figure 9-1 is an example, not the general case. In general, the Distributed Relay supports:

a) The necessary protocols and procedures for only the configurations listed in 9.3.1 as provided by this standard.
b) Link Aggregation functions, each subsuming one or more MACs.
c) Connections among the Portal Systems of the Distributed Relay.

The purpose of introducing the Distributed Relay functional layer in this example is to make these two Portal Systems appear, to the Systems connected to them, to be in the configuration illustrated in Figure 9-2. There appears to exist a third emulated System **C**, connected to the original Portal Systems by a link that has been inserted between Function 1 and Link Aggregation. That is, Portal Systems **A** and **B** conspire to behave, insofar as any other Systems to which they are connected can discern, as if emulated System **C**

actually exists, as shown in Figure 9-2. While Figure 9-2 is an example, it illustrates the principles of the Distributed Relay:

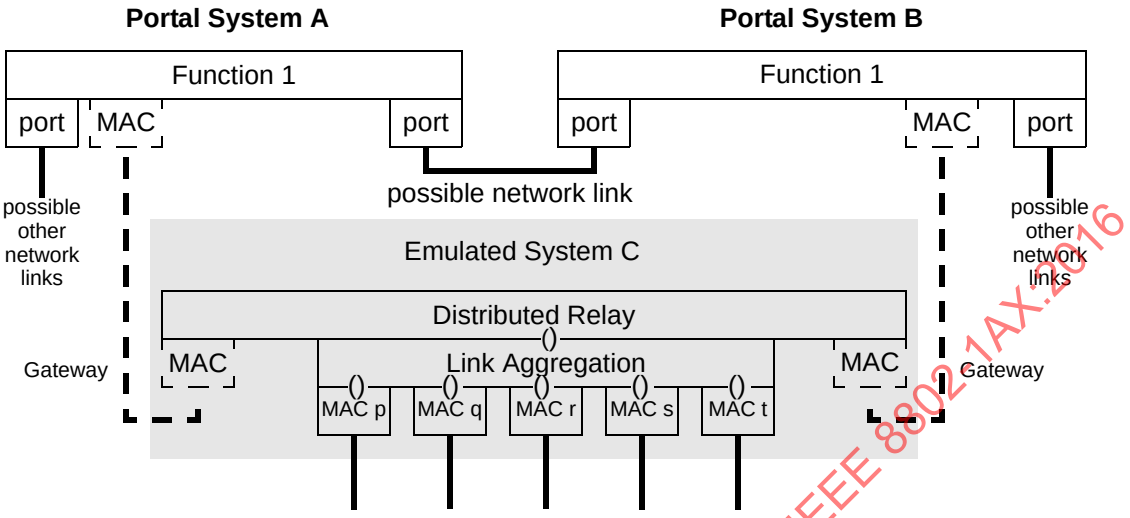


Figure 9-2—Distributed Relay: as seen by other Systems

d) The Distributed Relay in the emulated System C is an (N+1)-port relay for N Portal Systems, with N Gateway Ports connected to the Portal Systems, and a single Emulated Link Aggregation sublayer associated with the original Portal Systems.

e) The Aggregation Ports (MACs) have been moved to the emulated System, and thus appear, to all other Systems, to be equally distant from the real Portal Systems comprising the Distributed Relay.

The actual constructs used by the two Portal Systems to create the emulated System **C** are illustrated in Figure 9-1. Figure 9-1 shows the two DR Functions of the Distributed Relay, one DR Function in each System **A** and **B**. This example illustrates the remaining principles of the Distributed Relay:

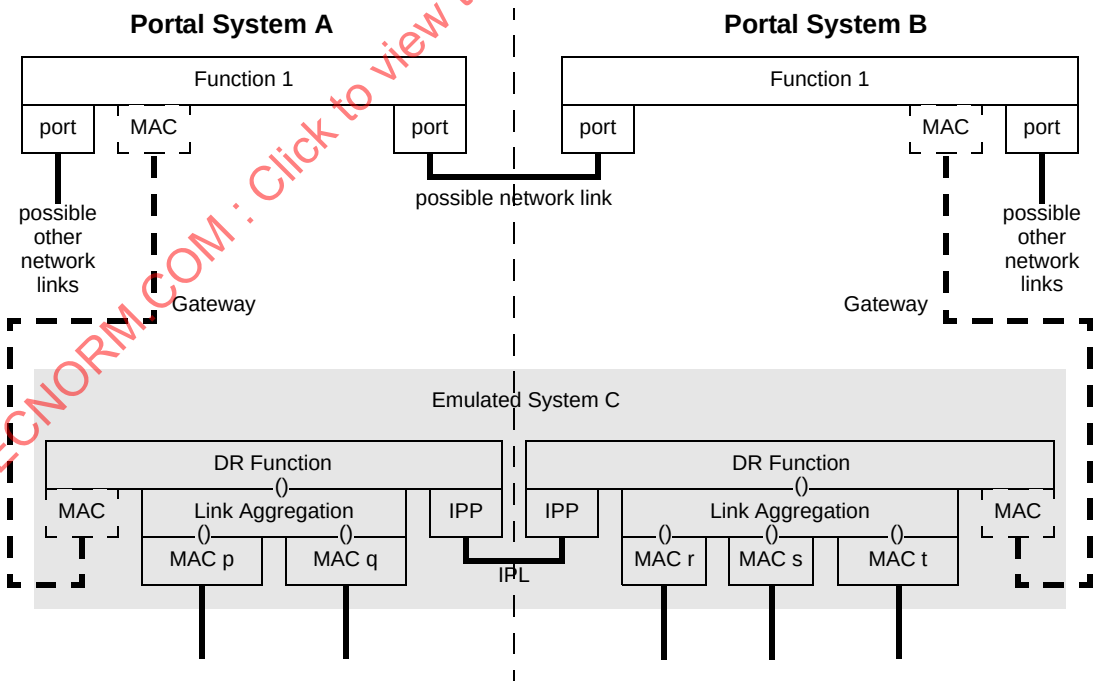


Figure 9-1—Distributed Relay: as seen by Systems A and B

In each System **A** and **B**, the ports that are to be associated with System **C** are moved to a position below the DR Function's Link Aggregation sublayer.

f) A virtual link and its terminating virtual MACs, called a "Gateway," is constructed to connect each DR Function to its Function 1.
g) Between each pair of DR Functions in the Portal there is constructed an Intra-Portal Link (IPL), terminated at each end by an Intra-Portal Port (IPP). (This can exist in many forms; see 9.3.2.)
h) There is a "Gateway algorithm" that decides through which Gateway a frame can pass into or out of the emulated Distributed Relay.
i) Similarly, a "Port algorithm" decides through which Portal System's Aggregation Port a frame can pass into or out of the emulated Distributed Relay.
j) As mentioned previously, there can be three Systems participating to create a Portal and a emulated System **C**. In that case, the Distributed Relay in emulated System **C** has an additional Gateway Port, one to each Portal System, and two or three IPLs to interconnect the DR Functions.
k) The DR Functions, as specified in 9.3, work together to move frames between the Gateways, the IPL, and the Link Aggregation sublayers.

NOTE 1—This standard does *not* define the alternate model shown in Figure 9-4, in which the entirety of Systems **A** and **B** simulate a single System **D**, but neither does this standard prevent the use of DRNI in such a model.

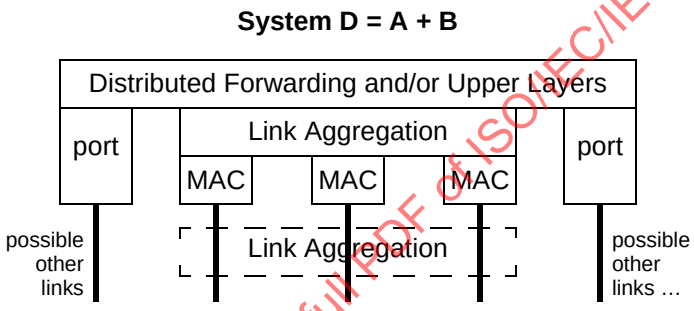


Figure 9-4—*Not an example* of a DRNI Portal

NOTE 2—The support of Figure 9-2 is important for achieving goal item e) in 9.1, administrative independence, without excessive standardization. System **C** in Figure 9-2 can be made visible to the network control protocols. This makes it relatively easy for Systems **A** and **B** to hide events occurring in the LAG or in the other Portal from the other Systems in their own network, because System **C** is connected only to Systems **A** and **B** in that network. Systems **A** and **B** maintain their own separate identities in their own network. They conspire to behave, insofar as any other Portals or Systems to which they are connected can discern, as if emulated System **C** as shown in Figure 9-2 actually exists, but if the Portals are in separately administered networks, this task is relatively easy. In the scheme shown in Figure 9-4, on the other hand, Systems **A** and **B** have to appear to be a single System **D** to all other Systems in their own network. This is, in general, a much more difficult task to accomplish, and one not easily standardized.

9.3 Distributed Relay operation and procedures

The DR Function in each Portal System (Figure 9-1) is intended to have (subject to operational failures) three kinds of ports:

a) Intra-Portal Ports (9.3.2), at most one (perhaps complex; see 9.3.2) IPL Port connected to each of the other Portal System(s) belonging to the same Portal;
b) Exactly one virtual Gateway Port with a virtual Link to a virtual Gateway Port in the Portal System in which the DR Function resides; and
c) Exactly one Aggregator Port (the port that is supported by the ISS instance identified by the prefix *Agg:* in 6.2.2) to the Link Aggregation sublayer with any number of Aggregation Ports, intended to connect to other Systems in a manner such that those other Systems believe they are connected to a single emulated System.

In Figure 9-2, the IPLs and IPPs are not visible, and the Distributed Relay of the emulated Aggregation System **C** has one Gateway to each of the Systems in its Portal.

The purpose of the emulated Distributed Relay is to pass every frame received from an Aggregator Port (Up frame) to a single Gateway, or to discard it, and every frame received from a Gateway (Down frame) to an Aggregator Port, or discard it. The DR Functions comprising the Distributed Relay sometimes has to send a frame across one or two IPLs in order to get it to the correct Gateway or Aggregator Port. A DR Function makes the choice of whether to discard or pass a frame to its Gateway, Aggregator Port, or one of its IPL Intra-Portal Links by assigning every frame to two Conversation IDs, a Gateway Conversation ID and a Port Conversation ID, and configuring the Gateways, Aggregation Ports, and IPL Intra-Portal Links in terms of these Conversation IDs.

The “Gateway algorithm” mentioned in item i) in 9.2 consists of two parts, an algorithm for assigning any given frame to a Gateway Conversation ID (7.4.1.1.13), and an assignment of Gateway Conversation IDs to Gateways (Drni_Gateway_Conversation, 9.3.4.2).

NOTE 1—If the Portal System is a VLAN Bridge performing learning, the mapping of a frame to a Gateway Conversation ID will be based on its VLAN ID to avoid oscillations in the learning process within the attached network. For implementations of DRNI in these cases there can be a one-to-one map of VLAN ID to Gateway Conversation ID.

Similarly, the “Port algorithm” of item j) in 9.2 consists of an algorithm for assigning any given frame to a Port Conversation ID (e.g., 8.2), and an assignment of Port Conversation IDs to Aggregation Ports (Drni_Port_Conversation, 9.3.4.2). The Port algorithm is identified by the aAggPortAlgorithm (7.3.1.1.33) and is also used in the operation of Conversation-sensitive frame collection and distribution described in 6.6.

Means are specified in 9.4 to ensure that all of the DR Functions on a given Portal use the same Gateway algorithm and the same Port algorithm for assigning frames to their respective Conversation IDs, and to guarantee that any given Gateway Conversation ID is assigned to at most one of the Gateways, and any given Port Conversation ID to at most one of the Aggregation Ports of a Portal, at any given moment.

It is allowed, but not required, that the Gateway algorithm and the Port algorithm use the same means for assigning a frame to a Conversation ID, so that the Gateway Conversation ID equals the Port Conversation ID.

NOTE 2—The Gateway selection process is in general dependent on the service parameters and forwarding capabilities of the hosting System while the Port selection process is dependent on the interface capabilities. For example a Bridge might use a Gateway algorithm based on VIDs while using a Port algorithm based on I-SIDs for its I-tagged interfaces.

The Port algorithm is always applied to a frame as it enters the DR Function from the Gateway to determine whether to send it to the Aggregator Port or to a particular IPP. The Gateway algorithm is always applied to a frame as it enters from the Aggregator Port to determine whether to send it to the Gateway or to a particular IPP. Both algorithms have to be applied, and their results compared, in order to forward a frame entering a DR Function from an IPL, as shown in Table 9-1.

Table 9-1—DR Function: forwarding frame received from IPL *n*

Gateway algorithm says, "Gateway is ..."	Port algorithm says, "Aggregation Port is ..."	DR Function emits frame to:
my Gateway	any[a]	my Gateway
behind IPL *n*	one of my Aggregation Ports	my Aggregator Port
	behind IPL *n*	discard[b]
	behind IPL *m* ($\neq n$)	IPL *m* ($\neq n$)
behind IPL *m* ($\neq n$)	any[a]	IPL *m* ($\neq n$)

[a] "Any" means that the output from the Port algorithm is not used; the Gateway algorithm determines to which port the frame is sent.
[b] DR Functions in different Portal Systems have incompatible configurations, or there is a malfunction. Discard frame to prevent looping.

The rationale for Table 9-1 is as follows:

d) Table 9-1 assumes one of three configurations (see 9.3.1):
   1) Two Portal Systems connected by a single IPL;
   2) Three Portal Systems connected linearly by two IPLs; or
   3) Three Portal Systems connected circularly by three IPLs.
e) The Gateway algorithm is enforced in both directions through the Gateway; that is, a frame entering the DR Function from a Gateway is discarded if the Gateway algorithm, applied to that frame, would not send it back up the Gateway. This is necessary to prevent frames received from a Gateway being forwarded across an IPL and passed back into the network through another Gateway.
f) If the Gateway algorithm indicates that the frame should pass through the Gateway, it has to be an Up frame, because a Down frame could not enter the Portal from any other DR Function, by item e).
g) Otherwise, if the Gateway algorithm indicates that the frame came from the IPL to which it would be forwarded, then it is a Down frame, and so is forwarded using the Port algorithm. (If the Port algorithm would send it back on the port on which it arrived, there is some sort of malfunction or misconfiguration, and the frame is discarded.)
h) Otherwise, the Gateway algorithm indicates that the frame did not come from the IPL to which it would be forwarded, so it has to be an Up frame, and the frame is directed according to the Gateway algorithm.

NOTE 3—The Port algorithm and DRCP of the Distributed Relay together determine the mapping of Port Conversation IDs to individual Aggregation Ports, and the variables in their control determine the operation of the Frame Distributor and Frame Collector. However, this does not alter the path of data and control as described in Clause 6, since the Distributed Relay passes all data to or from the Aggregation Ports through the Aggregator.

The application of the Gateway and Port algorithms on frames entering a DR Function from a Gateway and an Aggregator Port, as shown in Table 9-2 and Table 9-3, respectively.

Table 9-2—DR Function: forwarding frame received from my Gateway

Gateway algorithm says, "Gateway is ..."	Port algorithm says, "Aggregation Port is ..."	DR Function emits frame to:
my Gateway	one of my Aggregation Ports	my Aggregator Port
	behind IPL *n*	IPL *n*
behind IPL *n*	any	discard

.

Table 9-3—DR Function: forwarding frame received from one of my Aggregation Ports

Gateway algorithm says, "Gateway is ..."	Port algorithm says, "Aggregation Port is ..."	DR Function emits frame to:
my Gateway	any	my Gateway
behind IPL *n*	any	IPL *n*

9.3.1 Portal Topology

The most general topology of a Portal (Figure 9-5) could be the following:

a) Three Portal Systems connected in a ring by three Intra-Portal Links

Other supported topologies are subsets of this, including the following:

b) Three Portal Systems connected in a chain by two IPLs
c) Two Portal Systems connected by a single IPL
d) A Portal System with no active IPLs

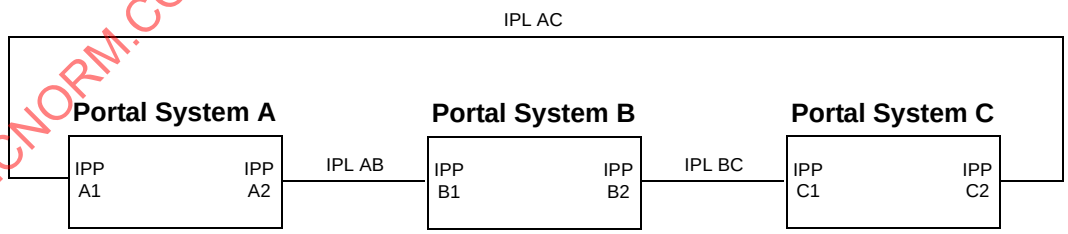


Figure 9-5—Portal Topology

These subsets can arise by design or as a result of system or link failures.

The terms Home, Neighbor, and Other neighbor are used to identify the Portal Systems from the point of view of a given Intra-Portal Port. The Home is the system containing the IPP. The Neighbor is the system

connected to the IPP. The Other neighbor is the system connected to the other IPP (if present) in the Home system. Using IPP B1 for an example, its home system is B, its immediate neighbor is A (because it is connected to IPP B1 via IPL AB), and its other neighbor is C (because it is connected to IPP B2 via IPL BC). Note that the Other neighbor of IPP A2 is also C (because it is connected to IPP A1 via IPL AC), so comparing the IDs of the Other neighbors of IPPs B1 and A2 verifies that there are no more than three systems in a ring or chain.

9.3.2 Intra-Portal Link

An Intra-Portal Link (IPL) is a single logical point-to-point link between DR Functions in two different Systems. A DR Function has at most one IPL for each of the other Systems comprising one end of a DRNI LAG. IPLs and network links can share a physical link or link aggregation.

An IPL can be physical (e.g., an IEEE 802.3 Ethernet LAN) or logical (e.g., an IEEE 802.1Q Backbone Service Instance or an IETF pseudowire). An Intra-Portal Link can share a physical link with other Intra-Portal Links or network links. An Intra-Portal Link can be a LAG, and thus consist of a number of physical links.

It will often be the case in deployed networks that two Systems will be configured with both a normal network link connecting them, and an IPL, as illustrated in Figure 9-1. It would reduce the utility of DRNI if every Portal required its own separate physical IPL, especially if a pair of Systems are configured to support multiple Portals. DRNI supports a number of methods by which the Systems can distinguish frames on a network link from frames on a particular IPL:

a) **Physical**. A separate physical link can be used to support any particular network link or IPL.
b) **Aggregated**. A separate Aggregator Port can be used for supporting an IPL.
c) **Time-shared**. A network link and one or more IPLs can use the same physical link (or Aggregator Port), but at different times. This requires that the Systems disable the use of the network link when the IPL is required for connectivity, or else that the use of the Aggregation Links and the selection of Gateways be adjusted to eliminate the need for using the IPL when the network link is required. This technique is described in 9.3.2.1.
d) **Tag-shared**. A network link and one or more IPLs can use the same physical link (or Aggregator Port), using different Service IDs. Tag sharing is described in 9.3.2.2.
e) **Logical**. The frames on the network link and the IPL(s) can be encapsulated, as described in 9.3.2.3.

A System implementing the DRNI shall support using separate physical links for IPLs and network links, and may support any of the other methods.

At each end of the shared physical link or Aggregator Port, there is one virtual port for each function (network link or a particular IPL) that the link or Aggregator Port is used for. Any of the preceding methods can be used simultaneously between two Systems, by administrative choice, as long as both ends of any given physical link or Aggregator Port use the same method.

9.3.2.1 Network / IPL sharing by time

The goal of Network/IPL sharing by time is to support DRNI without requiring separate physical links for a network connection and the IPL, and without requiring any frame modification. When sharing a link, it is necessary to be able to determine for each frame whether that frame is intended to be traversing the network connection or the IPL. This determination can be made without modifying the frame (e.g., without translating the VLAN ID or adding a tag or encapsulation) if at any given time the physical link is known to be only used as a network link or only used as an IPL. Whether the link is used as a network link or an IPL at any given time is established by the control protocol used to establish a fully connected, loop-free active topology for each VLAN in the network.

a) If the link is not included in the active topology of a VLAN (e.g., it has been blocked by the operation of the network control protocol), it is available to be used as the IPL. In this case the link is used by DRNI just as if it were a dedicated (unshared) IPL.
b) If the link is included in the active topology of a VLAN, then there is no IPL available for that VLAN. In this case the DRNI is unable to pass a frame between an Aggregation Port in one Portal System and a Gateway in another Portal System. Therefore for any given frame, DRNI is restricted to have the Gateway and the Aggregation Port in the same Portal System.

NOTE—The fact that the shared link can be unavailable for the transfer of frames between a Gateway and a specific Aggregation Port does not restrict the ability to exchange DRCPDUs on the shared link.

There are two cases to consider in meeting the restriction that for any given frame the Gateway and the Aggregation Port are in the same Portal System. The straightforward case is when the Port Conversation IDs are agreed and symmetric across the DRNI, and all frames with a given VLAN ID map to the Port Conversation IDs that select Aggregation Ports in the same Portal System. Then that Portal System is selected as the Gateway for that VLAN ID, and there is no need for any data frames to traverse the IPL. In any other circumstance the only way to assure that the Gateway and a specific Aggregation Port are in the same Portal System is that when a Portal System receives a frame on any port other than the shared network/IPL port, that Portal System is considered the Gateway for that frame and in each Portal System all Port Conversation IDs map to an Aggregation Port attached to that System. In this mode, a Portal System that receives any frame on a network port (not being the IPP or the Aggregation Port) is responsible for forwarding that frame to the Aggregation Port if required. A Portal System that receives a frame on the IPP never forwards that frame to the Aggregation Port. In this case the gateway selection cannot necessarily be based on VID, so when the Portal Systems are IEEE 802.1Q Bridges the learning process on the shared network/IPL link is compromised. Because the learning issues are limited to this port, it can be remedied by synchronizing the addresses learned on the DRNI with the other Portal System. See Annex G for further details on MAC Address Synchronization.

9.3.2.2 Network / IPL sharing by tag

If per-service frame distribution (8.2) is employed, and if the number of services required to support the network link, plus the number of services required to support one or more IPLs, is less than the number of services supplied by the frame format used (e.g., 4094 S-VLAN IDs), then VID translation can be used to separate the frames on the different logical links.

The method is selected by configuring the aDrniEncapsulationMethod (7.4.1.1.17) with the value 2. If enabled, as indicated by the variable Enabled_EncTag_Shared that is controlled by the Network/IPL sharing machine (9.4.20), every frame that is to be transported by the IPL to the Neighbor Portal System and is associated with a Gateway Conversation ID, is translated to use a value configured in aDrniIPLEncapMap (7.4.1.1.18) and every frame, that is to be transported by network link, shared with the IPL, and is associated with a Gateway Conversation ID, is translated to use a value configured in aDrniNetEncapMap (7.4.1.1.19).

9.3.2.3 Network / IPL sharing by encapsulation

This method enables sharing an IPL with a network link by using encapsulation techniques (e.g., an IEEE 802.1Q Backbone Service Instance, a B-VLAN, an IETF pseudowire, etc.)

The method is selected by configuring the aDrniEncapsulationMethod (7.4.1.1.17) with a value representing the encapsulation method that is used to transport IPL frames to the Neighbor Portal System when the IPL and network link are sharing the same physical link. This value consists of the 3-octet OUI or Company Identifier (CID) identifying the organization that is responsible for this encapsulation and one following octet used to identify the encapsulation method defined by that organization. If enabled, as indicated by the variable Enabled_EncTag_Shared that is controlled by the Network/IPL sharing machine (9.4.20), every frame, that is to be transmitted on the IPL to the Neighbor Portal System and is associated with a Gateway Conversation ID, is encapsulated with a method specified by aDrniEncapsulationMethod to use a value con-

figured in aDrniIPLEncapMap (7.4.1.1.18) and every frame, that is received by the IPL is de-encapsulated and mapped to a Gateway Conversation ID using the same table.

9.3.3 Protocol Identification

All DRNI protocols use the Distributed Resilient Network Interconnect EtherType value as the primary means of protocol identification; its value is shown in Table 9-4.

Table 9-4—DRNI EtherType Assignment

Assignment	Value
DRNI EtherType	89–52

The first octet of the MAC Client data following the EtherType field is a protocol subtype identifier that distinguishes between different DRNI protocols. Table 9-5 identifies the semantics of this subtype.

Table 9-5—DRNI Protocol subtypes

Protocol Subtype value	Protocol name
0	Reserved for future use
1	Distributed Relay Control Protocol (DRCP) (9.4)
2	Address Synchronization Protocol (Annex G)
3–255	Reserved for future use

9.3.4 DR Function state machines

The DR Function state machines shall implement the forwarding rules specified in Table 9-1, Table 9-2, and Table 9-3.These forwarding rules can be summarized as follows:

a) For frames entering through an Aggregation Port, Up frames, the Gateway algorithm decides whether to transmit it to the Gateway link or to the IPP according to its Gateway Conversation ID. If the frame's Gateway Conversation ID matches the Portal System's operational Gateway Conversation ID, the frame will be forwarded to the Gateway, otherwise it will be forwarded to the IPP.
b) For frames entering through a Gateway, Down frames, the Port algorithm decides whether to transmit it to the Aggregation Port or to the IPP according to its Port Conversation ID. If the frame's Port Conversation ID matches the Portal System's operational Port Conversation ID, the frame will be forwarded to the Aggregation Port, otherwise it will be forwarded to the IPP.
c) An Up frame offered to an IPP is only transmitted if the Portal System for this Gateway Conversation ID lies behind that IPP, otherwise it is discarded.
d) A Down frame offered to an IPP is only transmitted if the target Portal System for this Port Conversation ID lies behind that IPP, otherwise it is discarded

Some of the Link Aggregation variables used by a Distributed Relay have to be formed in a particular manner, so that multiple Portal Systems can cooperate to create a single emulated System, as follows:

e) The two least significant bits of the Port Priority in the Port ID (6.3.4) for each Aggregation Port in a Distributed Relay's Aggregator Port are set to the value of DRF_Portal_System_Number.
f) The most significant two bits of the Administrative Key for each Aggregation Port and associated Aggregator in a Distributed Relay's Aggregator Port are set to the value of DRF_Portal_System_Number. The remainder of the bits can be used as described in 6.3.5 to reflect the physical characteristics of the Aggregation Ports and they should be set consistently across the Portal Systems in a Portal so that a single Aggregator can be formed across the Distributed Relay.

The DR Function runs Link Aggregation with a single Aggregator. If the DR Function's Aggregation Ports can form more than one Link Aggregation Group, only one LAG can be attached to the DR Function's Aggregator (6.4.14.1).

9.3.4.1 Service interfaces

Since a DR Function uses various instances of the ISS, 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 interfaces is being invoked. The prefixes are as follows:

a) *Agg:*, for primitives issued on the interface between the DR Function and the Link Aggregation sublayer.
b) *Gate:*, for primitives issued on the Gateway.
c) *MacIppN:*, for primitives issued on the interface between the MAC entities supporting the IPL *n* and the DRCP Control Parser/Multiplexer (9.4.4).
d) *DRCPCtrlMuxN:*, for primitives issued on the interface between DRCP Control Parser/Multiplexer N and the DRCP control entity (where N is identifying the IPP associated with the DRCP Control Parser/Multiplexer).
e) *IppN:*, for primitives issued on the interface of the DR Function supported by the DRCP Control Parser/Multiplexer N (where N is identifying the IPP associated with the DRCP Control Parser/ Multiplexer).

9.3.4.2 Per-DR Function variables

DA
SA
mac_service_data_unit
priority
: The parameters of the M_UNITDATA.indication primitive.

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

Drni_Portal_System_Gateway_Conversation
: Operational Boolean vector, indexed by Gateway Conversation ID, indicating whether the indexed Gateway Conversation ID is allowed to pass through this DR Function's Gateway (TRUE = passes). Its values are computed by the updatePortalSystemGatewayConversation function (9.4.11).
Value: sequence of Boolean values, indexed by Gateway Conversation ID.

Drni_Portal_System_Port_Conversation
: Operational Boolean vector, indexed by Port Conversation ID, indicating whether the indexed Port Conversation ID is allowed to be distributed through this DR Function's Aggregator (TRUE = passes). Its values are computed by the updatePortalSystemPortConversation function (9.4.11).
Value: sequence of Boolean values, indexed by Port Conversation ID.

9.3.4.3 Per-IPP Intra-Portal Port variables

Ipp_Gateway_Conversation_Direction

Operational Boolean vector of which Gateway Conversation IDs are assigned to Gateways reachable through this IPP Intra-Portal Port. It is set by the operation of DRCP.

Value: Vector of Boolean flags indexed by Gateway Conversation ID; TRUE = some Gateway reachable through this IPP Intra-Portal Port is enabled for this Gateway Conversation ID.

Ipp_Gateway_Conversation_Direction is initialized to FALSE and recomputed by the updateIPPGatewayConversationDirection function (9.4.11) whenever any of its contributing variables changes. For frames received on this IPP Intra-Portal Port, TRUE means that the frame is a Down frame, ultimately destined for an Aggregator (or discard), and FALSE means the frame is an Up frame, ultimately destined for a Gateway (or discard). For frames offered for transmission on this IPP Intra-Portal Port, TRUE indicates that the frame can pass, and FALSE that it cannot.

Ipp_Port_Conversation_Passes

Operational Boolean vector of which Port Conversation IDs are allowed to be transmitted through this IPP Intra-Portal Port.

Value: Vector of Boolean flags indexed by Port Conversation ID.

This variable is examined only when a Down frame is offered for transmission on this IPP Intra-Portal Port. Ipp_Port_Conversation_Passes is initialized to FALSE and recomputed by the updateIPPPortConversationPasses function (9.4.11) whenever any of its contributing variables changes.

9.3.4.4 Functions

extractGatewayConversationID

This function extracts a Gateway Conversation ID value by applying the Gateway Algorithm (7.4.1.1.13) to the values of the parameters of the service primitive that is invoked on the DR Function’s Relay entity on receipt of an ISS primitive at one of the DR Function’s port. The relationship of the parameter values on the ISS primitives and the service primitives on the DR Function's Relay entity ports is provided by the associated supporting functions on those ports and their configuration.

NOTE—These supporting functions can be as simple as the EISS supporting functions specified in 6.9 of IEEE Std 802.1Q-2014 for the case of a DRNI supported on a Customer Network Port or a Provider Network Port on a Provider Bridge (Clause 15 in IEEE Std 802.1Q-2014), or more complex, like the EISS supporting functions specified in 6.10 or 6.11 in IEEE Std 802.1Q-2014, for the case of DRNI supported on a Provider Instance Port or a Customer Backbone Port, respectively, on a Backbone Edge Bridge (Clause 16 in IEEE Std 802.1Q-2014), or like the C-tagged Service Interface supporting functions or the Remote Customer Service Interface supporting functions specified in 15.4 or 15.6 in IEEE Std 802.1Q-2013 for the case of DRNI supported on a Customer Edge Port or a Remote Customer Access Port, respectively, on a Provider Edge Bridge.

Value: Integer in the range of 0 through 4095.

extractPortConversationID

This function extracts a Port Conversation ID value by applying the Port Algorithm (7.3.1.1.33) to the values of the parameters of the service primitive that is invoked on the Aggregator on receipt of a ISS primitive at one of the other DR Function’s port. The relationship of the parameter values on the ISS primitives on the Aggregator and the corresponding service primitives on the DR Function's port is provided by the associated supporting function on the Aggregator and the DR Function port and their configurations. Check the preceding NOTE.

Value: Integer in the range of 0 through 4095.

9.3.4.5 Messages

Agg:M_UNITDATA.indication
Gate:M_UNITDATA.indication
IppN:M_UNITDATA.indication
Agg:M_UNITDATA.request
Gate:M_UNITDATA.request
IppN:M_UNITDATA.request

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

If Network / IPL sharing by tag (9.3.2.2), or Network / IPL sharing by encapsulation (9.3.2.3) methods are used, the service primitives IppN:M_UNITDATA.indication and IppN:M_UNITDATA.request need to be manipulated through the operation of functions that are controlled by the Network/IPL sharing machine in 9.4.20.

9.3.4.6 DR Function: Aggregator Port reception state machine

The DR Function: Aggregator Port reception state machine shall implement the function specified in Figure 9-6 with its associated parameters (9.3.4.2 through 9.3.4.5). There is one DR Function: Aggregator Port reception state machine per Portal System and there are as many PASS_TO_IPP_N states as IPPs in a Portal System, each identified by the index *n*, The prefix "n." in the diagram is used to identify the specific IPP n that the associated variable is related to.

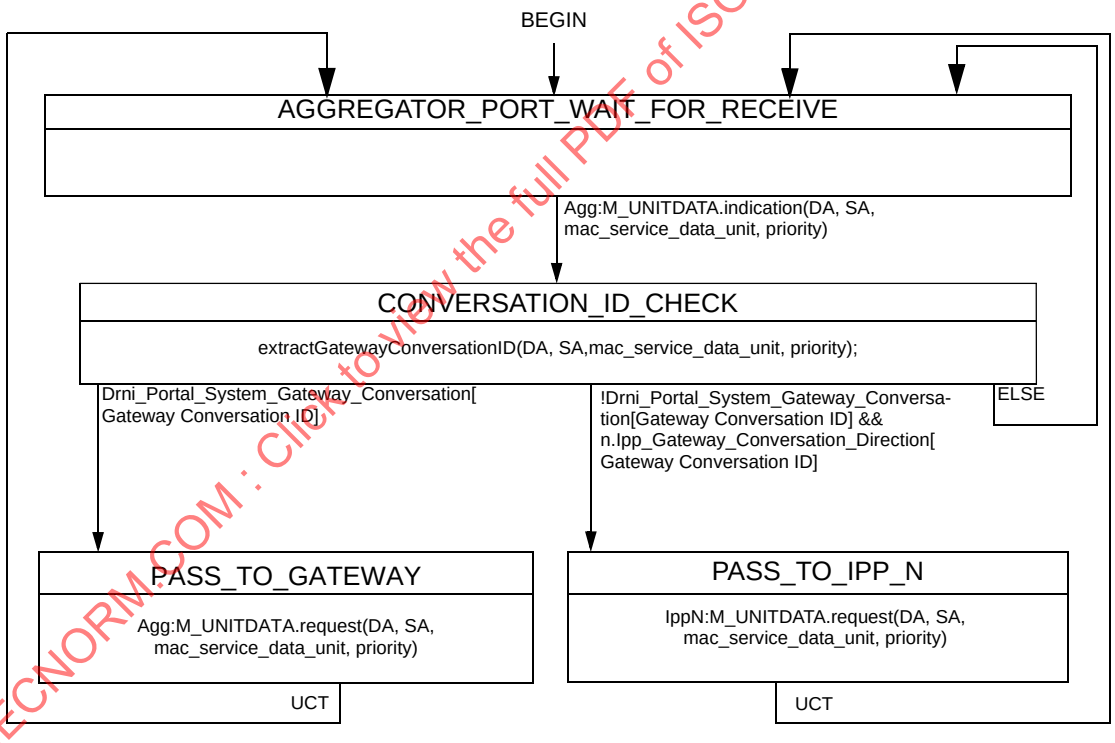


Figure 9-6—DR Function: Aggregator Port reception state machine

9.3.4.7 DR Function: Gateway distribution state machine

The DR Function: Gateway distribution state machine shall implement the function specified in Figure 9-7 with its associated parameters (9.3.4.2 through 9.3.4.5). There is one DR Function: Gateway distribution state machine per Portal System and there are as many PASS_TO_IPP_N states as IPPs in a Portal System,

each identified by the index *n*. The prefix "n." in the diagram is used to identify the specific IPP n that the associated variable is related to.

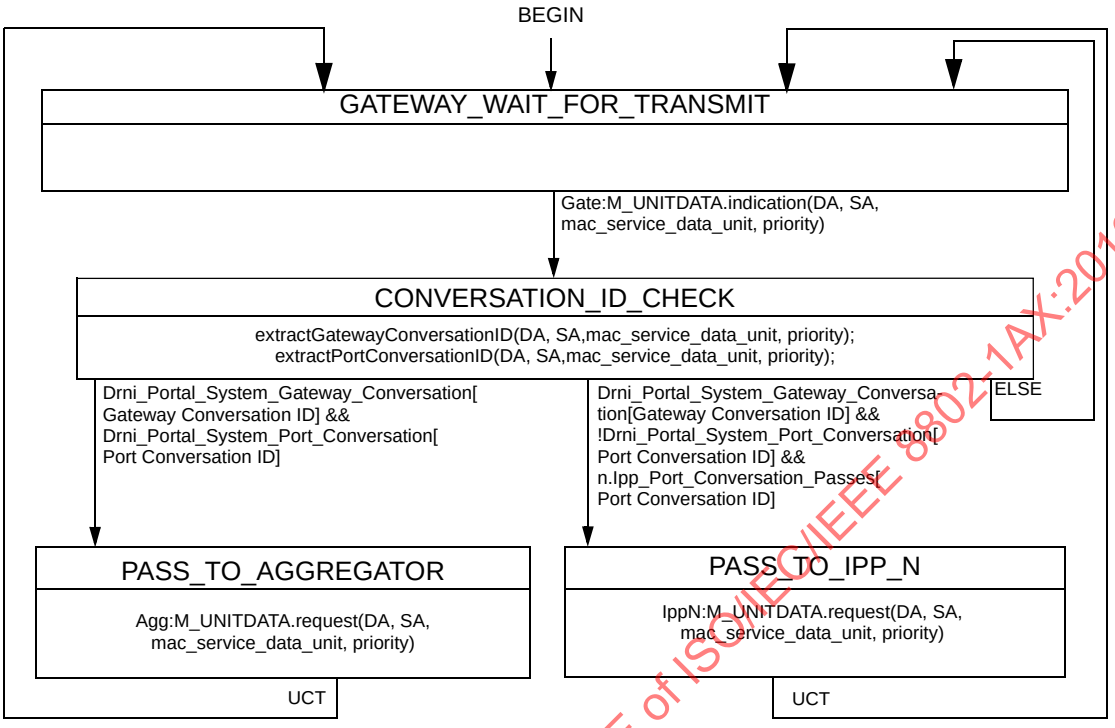


Figure 9-7—DR Function: Gateway distribution state machine

9.3.4.8 DR Function: IPP N reception state machine

The DR Function: IPP N reception state machine shall implement the function specified in Figure 9-8 with its associated parameters (9.3.4.2 through 9.3.4.5). There is one DR Function: IPP N reception state machine per IPP per Portal System and there are as many PASS_TO_IPP_M states as IPPs in a Portal System, each identified by the index *m*. The prefix "n." or "m." in the diagram is used to identify the specific IPP n or IPP m that the associated variable is related to.

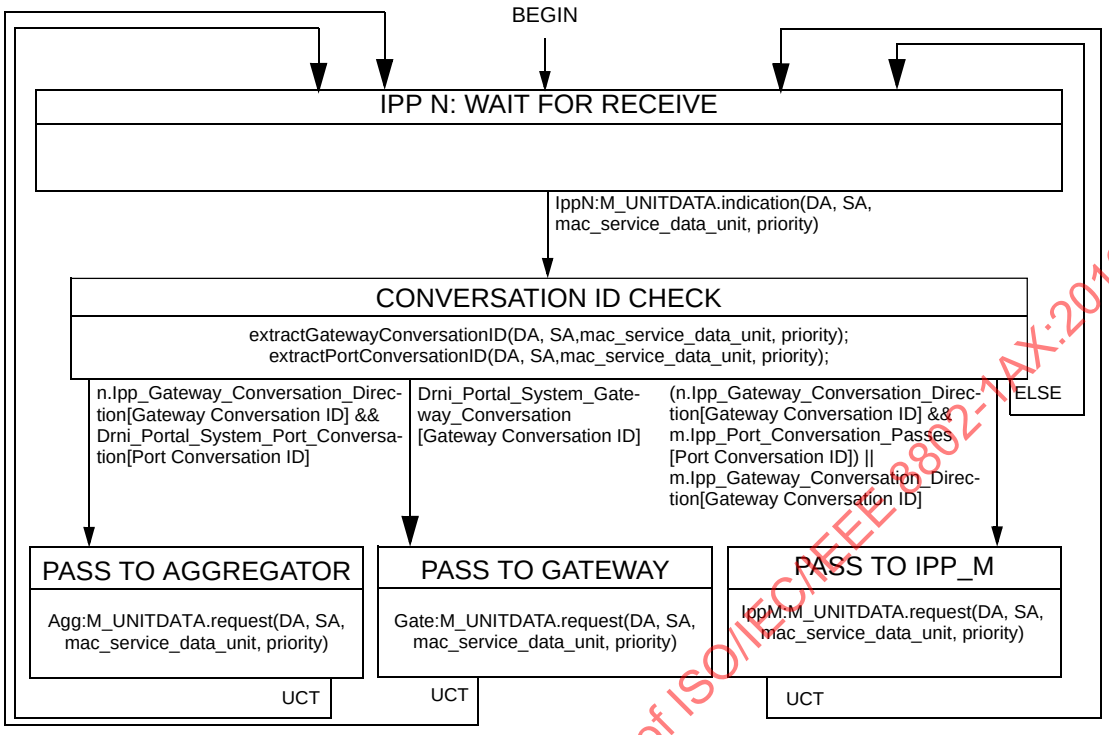


Figure 9-8—DR Function: IPP N reception state machine

9.4 Distributed Relay Control Protocol

The purpose of the DRCP is to:

a) Establish communication between Portal Systems across an Intra-Portal Link (9.4.1).
b) Verify the consistent configuration of Portal Systems (9.4.1).
c) Determine the identity to be used for the emulated System.
d) Distribute the current states of the Portal Systems and their Aggregation Ports among each other;
e) Compute the resultant path of any frame required to pass through each IPL Intra-Portal Link, and exchange the information with adjacent Portal Systems as required to ensure against forwarding loops and duplicate frame delivery.
f) Maintain Link Aggregation with a single Aggregator across the Portal.

The result of the operation of DRCP is to maintain the variables (9.3.4.2, 9.3.4.3) that control the forwarding of frames by the Distributed Relay.

DRCP exchanges information to ensure that the Portal Systems can work together. The first class of such information includes the managed objects and variables that have to be compatible in order to pass any data at all [item a)]. These include the following:

g) **aAggPortAlgorithm:** All Portal Systems have to use the same Port algorithm.
h) **aDrniGatewayAlgorithm:** All Portal Systems have to use the same Gateway algorithm.
i) **aDrniPortalAddr:** All Portal Systems have to have the same value for aDrniPortalAddr.
j) **aDrniPortalPriority:** All Portal Systems have to have the same value for aDrniPortalPriority.
k) **aDrniThreePortalSystem:** All Portal Systems have to have the same value for aDrniThreePortalSystem.

l) **aDrniPortalSystemNumber:** All Portal Systems have to have different aDrniPortalSystemNumber values, and all of these values have to be in the range 1...3.
m) **aAggActorAdminKey:** The most significant two bits of the Administrative Key for each Aggregator in a Distributed Relay's Aggregator Port is set to the value of DRF_Portal_System_Number. The remainder of the bits reflect the physical characteristics of the associated Aggregation Ports and they have to be the same for all Portal Systems in the Portal.

The second class of such information [item b)] includes managed objects that control through which Gateway and which Aggregation Ports each Conversation ID passes. For these managed objects, if the information about one Conversation ID is configured differently in different Portal Systems, only that Conversation ID is affected. Therefore, the Portal can operate normally, and the mechanisms that ensure against duplicate delivery or forwarding loops will block the passage of any frames belonging to misconfigured Conversation IDs. In order to detect misconfiguration, so that the blocking is not permanent, the DR Functions can notify the network administrator if the configurations differ. Since these configurations are quite large, a checksum of their contents is exchanged, rather than the configurations themselves. This method detects differences to a high probability, but not with complete certainty. These managed objects include the following:

n) **aDrniConvAdminGateway[]:** The list that is used to dynamically determine which Gateway Conversation ID flows through which Gateway.
o) **aAggConversationAdminLink[]:** The list that is used to dynamically determine which Port Conversation ID flows through which link on the LAG. It is the source of the equivalent mapping of Port Conversation IDs to Aggregation Ports used inside the Portal.

DRCP uses its information on which of the expected Portal Systems are either connected or are not connected via IPL Intra-Portal Links to determine the identity of the emulated Distributed Relay [item c)].

The current operational states of all of the Portal Systems and their Aggregation Ports is exchanged so that each of the DR Functions can determine to which Portal System's Gateway or Aggregator Port each frame is to be delivered [item d)]. Each DR Function computes a vector specifying exactly which Port Conversation IDs and which Gateway Conversation IDs can pass through each Gateway, Aggregation Port, or IPP Intra-Portal Port. On each IPP Intra-Portal Port, this information is exchanged [item e)]. Computation of the Gateway Conversation IDs that can pass through each Gateway requires knowledge at each Portal System in a Portal of the operational values of Gateway Vectors on every Gateway in the Portal. In order to avoid sending a 512 Octet vector for each of the Portal System in every DRCPDU exchange, a Gateway Vector database for every Gateway in a Portal is kept by every Portal System. This associates Gateway Vectors with Gateway Sequence numbers, enabling identification of a Gateway Vector in a DRCPDU exchange by its associated Gateway Sequence number. The Gateway Vector database for the Home Portal System should contain at least the two most recent entries. Only when the Gateway Vector changes the new values with its associated Gateway Sequence number would need to be included in a DRCPDU. If there is a difference between two DR Functions' vectors for any given Conversation ID, the output variables are set so that the DR Function will block frames with that Conversation ID. This prevents looping or duplicate delivery of any frame.

The values of the Keys to be used by the Aggregation Ports on each individual Portal System together with the associated received Operational Keys (DRF_Neighbor_Oper_Partner_Aggregator_Key, DRF_Other_Neighbor_Oper_Partner_Aggregator_Key) are exchanged among the Portal Systems so that a single Aggregator across the Portal is formed [item f)]. Only the Aggregation Ports that report a Partner_Oper_Key that is equal to the lowest numerical non zero value of the set comprising the values of the DRF_Home_Oper_Partner_Aggregator_Key, the DRF_Neighbor_Oper_Partner_Aggregator_Key, and the DRF_Other_Neighbor_Oper_Partner_Aggregator_Key, on each IPP on the Portal System are allowed to be attached to the Portal's Aggregator [item r) in 6.4.14.1].

9.4.1 Establishing the Portal and Distributed Relay

This standard does not specify the creation of Portals automatically. Instead, DRCP compares the network administrator's intentions, as defined by managed objects, to the physical topology of the configured Systems, and if the connected Systems' configurations are compatible, DRCP establishes and enables the operation of the Portal. In order to establish a Distributed Relay across a Portal, a network administrator configures the following managed objects:

a) There may be many Systems in a network, and some or all of the selected Portal Systems may participate in other Portals. Determining which other Portal Systems belong to this Portal System's Portal is accomplished by configuring aDrniPortalAddr (7.4.1.1.4) and aDrniPortalSystemNumber (7.4.1.1.7).
b) As described in 9.3.2, any point-to-point instance of the MAC Service can be assigned to be an Intra-Portal Link. The particular instances assigned to the use of a DR Function are configured in aDrniIntraPortalLinkList (7.4.1.1.8).
c) Which Aggregator in each Portal System is to be assigned to this DR Function is configured in aDrniAggregator (7.4.1.1.9).
d) The methods to be used by the DR Functions to assign frames to Gateway Conversation IDs and Port Conversation IDs are configured in two managed objects, aDrniGatewayAlgorithm (7.4.1.1.13) and aAggPortAlgorithm (7.3.1.1.33).
e) The prioritized list of assignments of Conversation IDs to Gateways and Aggregation Ports to cover failure modes are configured in several managed objects: aDrniConvAdminGateway[] (7.4.1.1.10) and aAggConversationAdminLink[] (7.3.1.1.35).
f) If the distributions methods used for the Gateway Conversation IDs and the Port Conversation IDs are the same, the aDrniPortConversationControl (7.4.1.1.23) managed object can be used to assign the Gateway Conversation IDs that are enabled to pass through a DR Function's Gateway based on the Port Conversation IDs assignment towards the DR Function's Aggregator Port, instead of the default operation which is to give the Gateway Conversation ID blocking to the control of the protocols running on the attached network.

9.4.2 DRCPDU transmission, addressing, and protocol identification

Distributed Relay Control Protocol Data Units (DRCPDUs) are transmitted and received using the service provided by an LLC entity that uses, in turn, a single instance of the MAC Service provided at an MAC Service Access Point (MSAP) associated with an IPP. Each DRCPDU is transmitted as a single MAC service request, and received as a single MAC service indication, with the following parameters:

a) destination address (9.4.2.1)
b) source address (9.4.2.2)
c) MAC Service Data Unit (MSDU)
d) priority (9.4.2.3)

The MSDU of each request and indication comprises a number of octets that provide EtherType protocol identification (9.4.2.4) followed by the DRCPDU proper (9.4.3).

NOTE 1—For the purposes of this standard, the term "LLC entity" includes entities that support protocol discrimination using the EtherType field as specified in IEEE Std 802.

NOTE 2—The complete format of an DRCP frame depends not only on the DRCPDU format, as specified in this clause, but also on the media access method dependent procedures used to support the MAC Service.

9.4.2.1 Destination MAC Address

The destination address (DA) for each MAC service request used to transmit a DRCPDU shall be one of the group address specified in Table 9-6 and selected by IPP Managed Objects Class (7.4.2).

Table 9-6—Distributed Relay Control Protocol destination addresses

Assignment	Value
Nearest Customer Bridge group address	01-80-C2-00-00-00
Nearest Bridge group address	01-80-C2-00-00-0E
Nearest non-TPMR Bridge group address	01-80-C2-00-00-03

Its default values shall be the Nearest non-TPMR Bridge group address.

9.4.2.2 Source MAC Address

The source address (SA) for each MAC service request used to transmit a DRCPDU shall be an individual address associated with the IPP MSAP at which the request is made.

9.4.2.3 Priority

The priority associated with each MAC Service request should be the default associated with the IPP MSAP.

9.4.2.4 Encapsulation of DRCPDUs in frames

A DRCPDU is encoded in the mac_service_data_unit parameter of an M_UNITDATA.request or M_UNITDATA.indication. The first octets of the mac_service_data_unit are a protocol identifier, followed by the DRCPDU, followed by padding octets, if any, as required by the underlying MAC service.

Where the ISS instance used to transmit and receive frames is provided by a media access control method that can support EtherType encoding directly (e.g., is an IEEE 802.3 MAC), the protocol identifier is two octets in length. All DRCPDUs are identified by the DRNI EtherType specified in Table 9-4 and the DRCP subtype specified in Table 9-5.

Where the ISS instance is provided by a media access method that cannot directly support EtherType encoding (e.g., is an IEEE 802.11 MAC), the EtherType protocol identifier is encoded according to the rule for a Subnetwork Access Protocol (Clause 9 of IEEE Std 802-2014) that encapsulates Ethernet frames over LLC, and comprises the SNAP header (hexadecimal AA-AA-03) followed by the SNAP PID (hexadecimal 00-00-00) followed by the DRNI protocol's EtherType (hexadecimal 89-52).

9.4.3 DRCPDU structure and encoding

9.4.3.1 Transmission and representation of octets

All DRCPDUs 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 DRCPDU 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.

9.4.3.2 DRCPDU structure

The DRCPDU structure shall be as shown in Figure 9-9 and as further described in the following field definitions:

a) *Subtype.* The Subtype field identifies the specific Protocol being encapsulated. DRCPDUs carry the Subtype value 0x01.
b) *Version Number.* This identifies the DRCP version; implementations conformant to this version of the standard carry the value 0x01.
c) ***TLV_type = Portal Information***. This field indicates the nature of the information carried in this TLV-tuple. DRNI information is identified by the integer value 1.
d) *Portal_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, Actor information uses a length value of 16.
e) *Aggregator_Priority.* The priority assigned to the Actor System ID (by management or administration policy) of the Aggregator attached to the DR Function, encoded as an unsigned integer from aAggActorSystemPriority (7.3.1.1.5).
f) *Aggregator_ID.* The MAC address component of Actor System ID of the Aggregator attached to the DR Function from aAggActorSystemID (7.3.1.1.4).
g) *Portal_Priority.* The priority assigned to the Portal's System ID (by management or administration policy), encoded as an unsigned integer from aDrniPortalPriority (7.4.1.1.5).
h) *Portal_Addr.* The MAC address component of Portal's System ID from aDrniPortalAddr (7.4.1.1.4).
i) ***TLV_type = Portal Configuration Information***. This field indicates the nature of the information carried in this TLV-tuple. *Portal Configuration Information* is identified by the integer value 2.
j) *Portal_Configuration_Information_Length.* This field indicates the length (in octets) of this TLV-tuple, *Portal Configuration Information* uses a length value of 43.
k) *Topology_State.* This DR Function's topology related variables for the IPP, encoded as individual bits within a single octet, as follows and as illustrated in Figure 9-10:
    1) *Portal_System_Number* is encoded in bits 0 and 1. The Portal System Number of this DR Function from aDrniPortalSystemNumber (7.4.1.1.7).
    2) *Neighbor_Conf_Portal_System_Number* is encoded in bits 2 and 3. The configured Portal System Number of the Portal System that is attached to this IPP (7.4.1.1.8).
    3) *3_System_Portal* is encoded in bit 4. This bit indicates if this is a Portal System that is part of a Portal consisting of three Portal Systems. It is encoded as 1 for a Portal of three Portal Systems and as 0 otherwise. It is always set equal to aDrniThreePortalSystem (7.4.1.1.6).
    4) *Common_Methods* is encoded in bit 5. It is encoded as 1 when aDrniPortConversationControl (7.4.1.1.23) is set to TRUE and as 0 otherwise.
    5) Bit 6 is reserved for future use. It is set to 0 on transmit and they are ignored on receipt.
    6) *Other_Non_Neighbor* (ONN, 9.4.9) is encoded in bit 7. TRUE (encoded as 1) indicates that the Other Ports Information TLV is not associated with an immediate Neighbor of this Portal System. FALSE (encoded as 0) indicates that the Other Ports Information TLV is an immediate Neighbor on the other IPP on this Portal System.This bit is only used if the Portal Topology contains three Portal Systems. If not, it is transmitted as 0 and ignored on receipt.
l) *Oper_Aggregator_Key.* The current operational Aggregator Key value of the Aggregator attached to the DR Function.

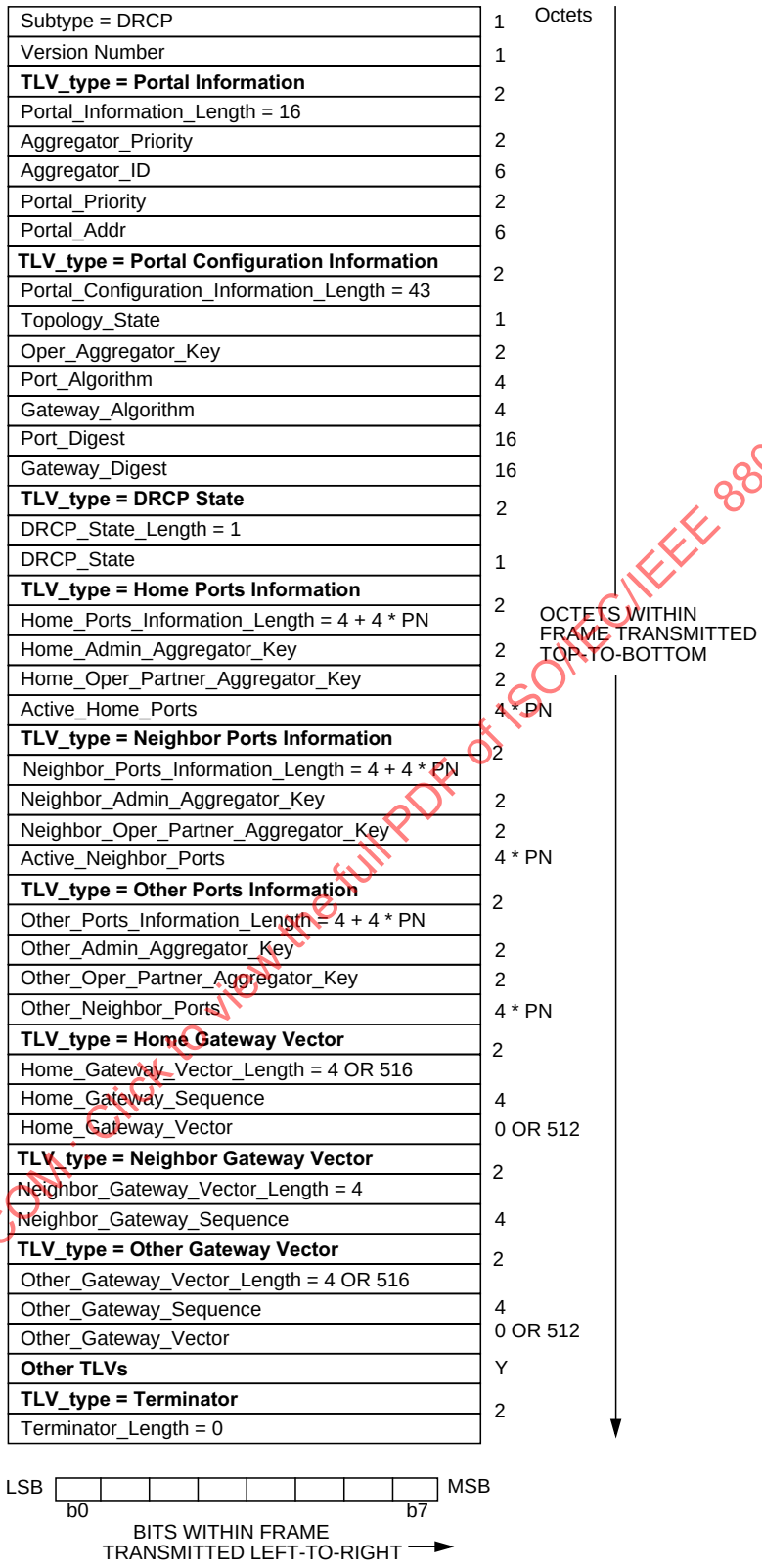
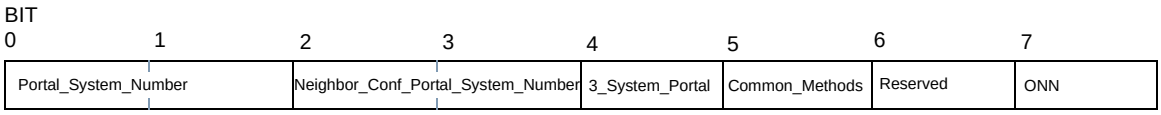


Figure 9-9—DRCPDU structure



NOTE 1—Bit ordering within this field is as specified in 9.4.3.1.

Figure 9-10—Bit encoding of the Topology_State field

m) *Port_Algorithm.* The Port algorithm used by this DR Function and Aggregator from aAggPortAlgorithm (7.3.1.1.33). Table 6-4 provides the IEEE 802.1 OUI (00-80-C2) Port Algorithm encodings.

n) *Gateway_Algorithm.* The Gateway algorithm used by this DR Function from aDrniGatewayAlgorithm (7.4.1.1.13). Table 9-7 provides the IEEE 802.1 OUI (00-80-C2) Gateway Algorithm encodings.

Table 9-7—IEEE Gateway algorithms

Gateway Algorithm field	Value
Reserved for IEEE 802.1	0
Distribution based on C-VIDs	1
Distribution based on S-VIDs	2
Distribution based on I-SIDs	3
Distribution based on TE-SIDs	4
Distribution based on ECMP Flow Hash	5
Reserved	6–255

o) *Port_Digest.* The DRF_Home_Conversation_PortList_Digest of this DR Function's prioritized Port Conversation ID-to-Link Number ID assignments from aAggConversationAdminLink[] (7.3.1.1.35).

p) *Gateway_Digest.* The DRF_Home_Conversation_GatewayList_Digest of this DR Function's prioritized Gateway Conversation ID- to-Gateway assignments from aDrniConvAdminGateway[] (7.4.1.1.10).

q) ***TLV_type = DRCP State.*** This field indicates the nature of the information carried in this TLV-tuple. *DRCP State* is identified by the integer value 6.

r) *DRCP_State_Length.* This field indicates the length (in octets) of this TLV-tuple, *DRCP State* uses a length value of 1.

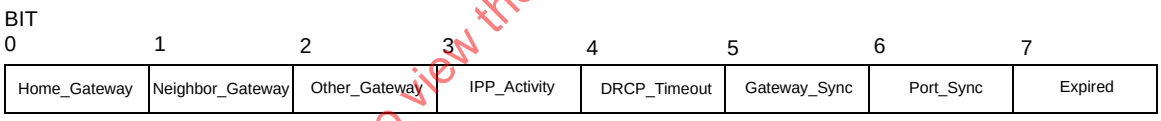
s) *DRCP_State.* This DR Function's DRCP variables for the IPP, encoded as individual bits within a single octet, as follows and as illustrated in Figure 9-11:

1) *Home_Gateway* is encoded in bit 0. This flag indicates the operational state of this DR Function's Gateway. TRUE indicates operational and is encoded as a 1; Non operational is encoded as a 0.

2) *Neighbor_Gateway* is encoded in bit 1. This flag indicates the operational state of the Neighbor DR Function's Gateway. TRUE indicates operational and is encoded as a 1; Non operational is encoded as a 0.

3) *Other_Gateway* is encoded in bit 2. This flag indicates the operational state of a potential Other neighbor DR Function's Gateway. TRUE indicates operational and is encoded as a 1; Otherwise the flag is encoded as a 0. This bit is only used if the Portal Topology contains three Portal Systems. If not, it is transmitted as 0 and ignored on receipt.
4) *IPP_Activity* is encoded in bit 3. This flag indicates the Neighbor Portal System's DRCP Activity on this IPP. Active DRCP Neighbor is encoded as a 1; No DRCP activity is encoded as 0.
5) *DRCP_Timeout* is encoded in bit 4. 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.
6) *Gateway_Sync* is encoded in bit 5. If TRUE (encoded as a 1), this DR Function considers this IPP's Neighbor Portal Systems to have their Gateways *IN_SYNC*; i.e., this Portal System's operational vector listing which Portal System's Gateway (if any) is passing each Gateway Conversation ID is in agreement with this IPP's Neighbor Portal Systems' operational vector. If FALSE (encoded as a 0), then this IPP is currently *OUT_OF_SYNC;* i.e., this IPP's Neighbor Portal Systems' operational vector listing which Portal System's Gateway (if any) is passing each Gateway Conversation ID is in disagreement.
7) *Port_Sync* is encoded in bit 6. If TRUE (encoded as a 1), this DR Function considers this IPP's Neighbor Portal Systems to have their Aggregator Ports *IN_SYNC*; i.e., this Portal System's operational vector listing which Portal System's Aggregation Port (if any) is passing each Port Conversation ID is in agreement with this IPP's Neighbor Portal Systems' operational vector. If FALSE (encoded as a 0), then this IPP is currently *OUT_OF_SYNC;* i.e., this IPP's Neighbor Portal Systems' operational vector listing which Portal System's Aggregation Port (if any) is passing each Port Conversation ID is in disagreement.
8) *Expired* is encoded in bit 7. If TRUE (encoded as a 1), this flag indicates that the DR Function's DRCPDU Receive machine is in the EXPIRED or DEFAULTED state; if FALSE (encoded as a 0), this flag indicates that the DR Function's DRCPDU Receive machine is neither in the EXPIRED nor DEFAULTED state.

NOTE 2—The received value of Expired state is not used by DRCP; however, knowing its value can be useful when diagnosing protocol problems.



NOTE 3—Bit ordering within this field is as specified in 9.4.3.1.

Figure 9-11—Bit encoding of the DRCP_State field

t) ***TLV_type = Home Ports Information***. This field indicates the nature of the information carried in this TLV-tuple. Home Ports information is identified by the integer value 4.
u) *Home_Ports_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Home Ports information uses a length value of 4 plus 4 times the number of this Portal System's Aggregation Ports that are included in this TLV.
v) *Home_Admin_Aggregator_Key.* The administrative Aggregator Key value of the Aggregator attached to this DR Function from aAggActorAdminKey (7.3.1.1.7).
w) *Home_Oper_Partner_Aggregator_Key.* The operational Partner Aggregator Key associated with this Portal System's Aggregator LAG ID.
x) *Active_Home_Ports.* The list of active Aggregation Ports in this Portal System, provided by the Drni_Portal_System_State[] variable, in increasing Port ID (6.3.4) order. The list is controlled by the operation of LACP (listing all the Ports on this Portal System for which LACP is declaring Actor_Oper_Port_State.Distributing = TRUE).
y) ***TLV_type = Neighbor Ports Information***. This field indicates the nature of the information carried in this TLV-tuple. Neighbor Ports information is identified by the integer value 5.

z) *Neighbor_Ports_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Neighbor Ports information uses a length value of 4 plus 4 times the number of the Neighbor Aggregation Ports that are included in this TLV.

aa) *Neighbor_Admin_Aggregator_Key.* The administrative Aggregator Key value of the Aggregator attached to the Neighbor Portal System.

ab) *Neighbor_Oper_Partner_Aggregator_Key.* The operational Partner Aggregator Key associated with the Neighbor Portal System's Aggregator LAG ID.

ac) *Active_Neighbor_Ports.* The list of active Aggregation Ports in the immediate Neighbor Portal System on this IPP, provided by the Drni_Portal_System_State[] variable, in increasing Port ID (6.3.4) order. The list is controlled by the operation of LACP (listing all the Ports on the immediate Neighbor Portal System for which LACP is declaring Actor_Oper_Port_State.Distributing = TRUE).

ad) ***TLV_type = Other Ports Information***. This field indicates the nature of the information carried in this TLV-tuple. The Other Ports information is identified by the integer value 6. This TLV is only used if the Portal Topology contains three Portal Systems.

ae) *Other_Ports_Information_Length.* This field indicates the length (in octets) of this TLV-tuple. Other Ports information uses a length value of 4 plus 4 times the number of the Other Portal System's Aggregation Ports that are included in this TLV.

af) *Other_Admin_Aggregator_Key.* The administrative Aggregator Key value of the Aggregator attached to the Other neighbor Portal System.

ag) *Other_Oper_Partner_Aggregator_Key.* The operational Partner Aggregator Key associated with the Other neighbor Portal System's Aggregator LAG ID.

ah) *Other_Neighbor_Ports.* The list of active Aggregation Ports in the Other neighbor Portal System associated with this IPP, provided by the Drni_Portal_System_State[] variable, in increasing Port ID (6.3.4) order. The list is controlled by the operation of LACP (listing all the Ports on an optional other Portal System for which LACP is declaring Actor_Oper_Port_State.Distributing = TRUE).

ai) ***TLV_type = Home Gateway Vector.*** This field indicates the nature of the information carried in this TLV-tuple. Home Gateway Vector is identified by the integer value 7.

aj) *Home_Gateway_Vector_Length.* This field indicates the length (in octets) of this TLV-tuple. Home Gateway Vector uses a length value of 4 or 516, the latter when the Home Gateway Vector field is included in the TLV.

ak) *Home_Gateway_Sequence.* The DRF_Home_Gateway_Sequence (9.4.8) sequence number associated with the operational Home Gateway Vector. This corresponds to the first entry in the Home Gateway database.

al) *Home_Gateway_Vector.* This field contains the value of the DRF_Home_Gateway_Conversation_Mask (9.4.8) Boolean vector indexed by increasing Gateway Conversation ID order starting from Gateway Conversation ID 0 up to Gateway Conversation ID 4095.

am) ***TLV_type = Neighbor Gateway Vector.*** This field indicates the nature of the information carried in this TLV-tuple. Neighbor Gateway Vector is identified by the integer value 8.

an) *Neighbor_Gateway_Vector_Length.* This field indicates the length (in octets) of this TLV-tuple. Neighbor Gateway Vector uses a length value of 4.

ao) *Neighbor_Gateway_Sequence.* The DRF_Neighbor_Gateway_Sequence (9.4.9) sequence number associated with the operational Neighbor Gateway Vector.

ap) ***TLV_type = Other Gateway Vector.*** This field indicates the nature of the information carried in this TLV-tuple. Other Gateway Vector is identified by the integer value 9. This TLV is only used if the Portal Topology contains three Portal Systems.

aq) *Other_Gateway_Vector_Length.* This field indicates the length (in octets) of this TLV-tuple. Other Gateway Vector uses a length value of 4 or 516, the latter when the Other Gateway Vector field is included in the TLV.

ar) *Other_Gateway_Sequence.* The DRF_Other_Neighbor_Gateway_Sequence (9.4.9) sequence number associated with the operational Home Gateway Vector.

as) *Other_Gateway_Vector.* This field contains the value of the DRF_Other_Neighbor_Gateway_Conversation_Mask (9.4.9) Boolean vector indexed by increasing Gateway Conversation ID order starting from Gateway Conversation ID 0 up to Gateway Conversation ID 4095.

at) ***Other TLVs***. This field contains additional optional TLVs (e.g., 9.4.3.4, 9.4.3.5).

au) ***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 0.

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

NOTE 4—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.

NOTE 5—The 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.

NOTE 6—The DRCPDU grows in size with the number of Aggregation Ports. A maximum of 90 Aggregation Ports spread across the Portal's Portal Systems are supported. The minimum number of Aggregation Ports that need to be supported by a Portal is two.

The basic TLV format, applicable to all DRCPDU TLVs, is shown in Figure 9-12.

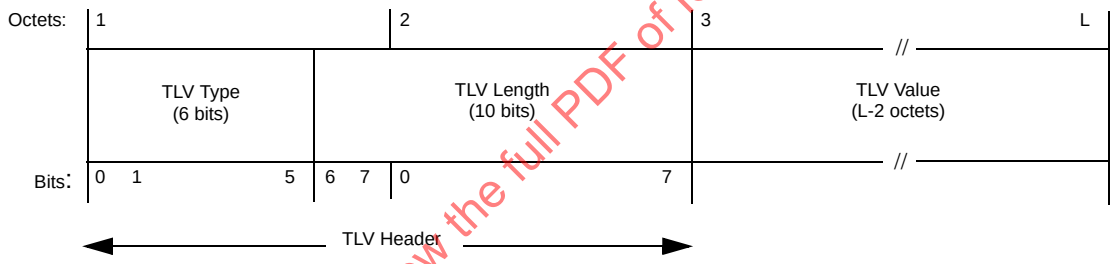


Figure 9-12—Basic TLV format

The TLV type field occupies the six least significant bits of the first octet of the TLV format. The two most significant bits in the first octet of the TLV format is the two least significant bits of the TLV length field.

NOTE 7—The DRCP TLV Length fields provide the length of the TLV Value fields in contrast to the LACP TLV Length fields that provide the length of the total TLV (including the 2 octets for the TLV Type and TLV Length field). This enables the use of TLVs carrying a TLV Value field of a length up to 1023 octets.

Table 9-8 provides a list of the TLVs that are applicable for DRCP.

Table 9-8—Type field values of DRCP TLVs

Type field	TLV	Support
0	Terminator TLV	Mandatory
1	Portal Information TLV	Mandatory
2	Portal Configuration Information TLV	Mandatory
3	DRCP State TLV	Mandatory
4	Home Ports Information TLV	Mandatory
5	Neighbor Ports Information TLV	Mandatory
6	Other Ports TLV	Mandatory only for Portals of 3 Portal Systems
7	Home Gateway Vector TLV	Mandatory
8	Neighbor Gateway Vector TLV	Mandatory
9	Other Gateway Vector TLV	Mandatory only for Portals of 3 Portal Systems
10	2P Gateway Conversation Vector TLV	Mandatory (see 9.4.3.3 for details)
11	3P Gateway Conversation Vector-1 TLV	Mandatory (see 9.4.3.3 for details)
12	3P Gateway Conversation Vector-2 TLV	Mandatory (see 9.4.3.3 for details)
13	2P Port Conversation Vector TLV	Mandatory (see 9.4.3.3 for details)
14	3P Port Conversation Vector-1 TLV	Mandatory (see 9.4.3.3 for details)
15	3P Port Conversation Vector-2 TLV	Mandatory (see 9.4.3.3 for details)
16	Network/IPL Sharing Method TLV	Optional (see 9.4.3.4 for details)
17	Network/IPL Sharing Encapsulation TLV	Optional (see 9.4.3.4 for details)
18	Organization Specific TLV	Optional (see 9.4.3.5 for details)
19–63	Reserved for IEEE 802.1	

9.4.3.3 Conversation Vector TLVs

Table 9-9 provides the list of the Conversation Vector TLVs. These TLVs shall be present in a DRCPDU only when certain conditions are met. In particular the2P Gateway Conversation Vector TLV shall only be present when GatewayConversationTransmit == TRUE and aDrniThreePortalSystem == FALSE, the 3P Gateway Conversation Vector-1 TLV accompanied by the 3P Gateway Conversation Vector-2 TLV shall only be present when GatewayConversationTransmit == TRUE and aDrniThreePortalSystem == TRUE, the 2P Port Conversation Vector TLV shall only be present when PortConversationTransmit == TRUE and aDrniThreePortalSystem == FALSE, and the 3P Port Conversation Vector-1 TLV accompanied by the 3P Port Conversation Vector-2 TLV shall only be present when PortConversationTransmit == TRUE and aDrniThreePortalSystem == TRUE. See the DRCPDU Transmit machine (9.4.19) for details regarding the presence of the Conversation Vector TLVs in a DRCPDU.

Table 9-9—Type field values of Conversation Vector TLVs

TLV	Type field
2P Gateway Conversation Vector TLV	10
3P Gateway Conversation Vector-1 TLV	11
3P Gateway Conversation Vector-2 TLV	12
2P Port Conversation Vector TLV	13
3P Port Conversation Vector-1 TLV	14
3P Port Conversation Vector-2 TLV	15

9.4.3.3.1 2P Gateway Conversation Vector TLV

The 2P Gateway Conversation Vector TLV structure shall be as shown in Figure 9-13 and as further described in the following field definitions:

TLV_type = 2P Gateway Conversation Vector	2
2P_Gateway_Conversation_Vector_Length = 512	
2P_Gateway_Conversation_Vector	512

Figure 9-13—2P Gateway Conversation Vector TLV

a) *TLV_type = 2P Gateway Conversation Vector.* This field indicates the nature of the information carried in this TLV-tuple. *2P Gateway Conversation Vector* is identified by the integer value 10.

b) *2P_Gateway_Conversation_Vector_Length.* This field indicates the length (in octets) of this TLV-tuple. *2P Gateway Conversation Vector* uses a length value of 512.

c) *2P_Gateway_Conversation_Vector.* The field provides the operational Drni_Portal_System_Gateway_Conversation (9.3.4.2) Boolean vector. 2P_Gateway_Conversation_Vector is encoded as a 512-octet Boolean vector, indexed by Gateway Conversation ID, based on the Drni_Portal_System_Gateway_Conversation. The first bit indicates if Gateway Conversation ID 0 is allowed to pass through this DR Function's Gateway, the second bit indicates if Gateway Conversation ID 1 is allowed to pass through this DR Function's Gateway, and so on until the final bit, which indicates if Gateway Conversation ID 4095 is allowed to pass through this DR Function's Gateway.

9.4.3.3.2 3P Gateway Conversation Vector-1 TLV

The 3P Gateway Conversation Vector-1 TLV structure shall be as shown in Figure 9-14 and as further described in the following field definitions:

TLV_type = 3P Gateway Conversation Vector-1	2
3P_Gateway_Conversation_Vector_1_Length = 512	
3P_Gateway_Conversation_Vector-1	512

Figure 9-14—3P Gateway Conversation Vector-1 TLV

a) *TLV_type = 3P Gateway Conversation Vector-1*. This field indicates the nature of the information carried in this TLV-tuple. *3P Gateway Conversation Vector-1* is identified by the integer value 11.
b) *3P_Gateway_Conversation_Vector_1_Length*. This field indicates the length (in octets) of this TLV-tuple. *3P Gateway Conversation Vector-1* uses a length value of 512.
c) *3P_Gateway_Conversation_Vector-1*. The field provides the operational Drni_Gateway_Conversation (9.4.7) vector listing which Portal System's Gateway (if any) is passing each of the first 2048 Gateway Conversation IDs (Gateway Conversation ID 0 to Gateway Conversation ID 2047). *3P_Gateway_Conversation_Vector-1* is encoded as a 512-octet vector as sequence of Portal System Numbers (0 for none), indexed by Gateway Conversation ID, based on the Drni_Gateway_Conversation. The first two bits provide the Portal System Number of the Portal System that is passing Gateway Conversation ID 0, the second two bits provide the Portal System Number of the Portal System that is passing Gateway Conversation ID 1, and so on until the final two bits, which provide the Portal System Number of the Portal System that is passing Gateway Conversation ID 2047.

9.4.3.3.3 3P Gateway Conversation Vector-2 TLV

The 3P Gateway Conversation Vector-2 TLV structure shall be as shown in Figure 9-15 and as further described in the following field definitions:

TLV_type = 3P Gateway Conversation Vector-2	2
3P_Gateway_Conversation_Vector_2_Length = 512	
3P_Gateway_Conversation_Vector-2	512

Figure 9-15—3P Gateway Conversation Vector-2 TLV

a) *TLV_type = 3P Gateway Conversation Vector-2*. This field indicates the nature of the information carried in this TLV-tuple. *3P Gateway Conversation Vector-2* is identified by the integer value 12.
b) *3P_Gateway_Conversation_Vector_2_Length*. This field indicates the length (in octets) of this TLV-tuple. *3P Gateway Conversation Vector-2* uses a length value of 512.
c) *3P_Gateway_Conversation_Vector-2*. The field provides the operational Drni_Gateway_Conversation (9.4.7) vector listing which Portal System's Gateway (if any) is passing each of the last 2048 Gateway Conversation IDs (Gateway Conversation ID 2048 to Gateway Conversation ID 4095). *3P_Gateway_Conversation_Vector-2* is encoded as a 512-octet vector as sequence of Portal System Numbers (0 for none), indexed by Gateway Conversation ID, based on the Drni_Gateway_Conversation. The first two bits provide the Portal System Number of the Portal System that is passing Gateway Conversation ID 2048, the second two bits provide the Portal System Number of the Portal System that is passing Gateway Conversation ID 2049, and so on until the final two bits, which provide the Portal System Number of the Portal System that is passing Gateway Conversation ID 4095.

9.4.3.3.4 2P Port Conversation Vector TLV

The 2P Port Conversation Vector TLV structure shall be as shown in Figure 9-16 and as further described in the following field definitions:

TLV_type = 2P Port Conversation Vector	2
2P_Port_Conversation_Vector_Length = 512	
2P_Port_Conversation_Vector	512

Figure 9-16—2P Port Conversation Vector TLV

a) *TLV_type = 2P Port Conversation Vector.* This field indicates the nature of the information carried in this TLV-tuple. *2P Port Conversation Vector* is identified by the integer value 13.
b) *2P_Port_Conversation_Vector_Length.* This field indicates the length (in octets) of this TLV-tuple. *2P Port Conversation Vector* uses a length value of 512.
c) *2P_Port_Conversation_Vector.* The field provides the operational Drni_Portal_System_Port_Conversation (9.3.4.2) Boolean vector. *2P_Port_Conversation_Vector* is encoded as a 512-octet Boolean vector, indexed by Port Conversation ID, based on the Drni_Portal_System_Port_Conversation. The first bit indicates if Port Conversation ID 0 is allowed to pass through this DR Function's Aggregator, the second bit indicates if Port Conversation ID 1 is allowed to pass through this DR Function's Aggregator, and so on until the final bit, which indicates if Port Conversation ID 4095 is allowed to pass through this DR Function's Aggregator.

9.4.3.3.5 3P Port Conversation Vector-1 TLV

The 3P Port Conversation Vector-1 TLV structure shall be as shown in Figure 9-17 and as further described in the following field definitions:

TLV_type = 3P Port Conversation Vector-1	2
3P_Port_Conversation_Vector_1_Length = 512	
3P_Port_Conversation_Vector-1	512

Figure 9-17—3P Port Conversation Vector-1 TLV

a) *TLV_type = 3P Port Conversation Vector-1*. This field indicates the nature of the information carried in this TLV-tuple. *3P Port Conversation Vector-1* is identified by the integer value 14.
b) *3P_Port_Conversation_Vector_1_Length.* This field indicates the length (in octets) of this TLV-tuple. *3P Port Conversation Vector-1* uses a length value of 512.
c) *3P_Port_Conversation_Vector-1*. The field provides the operational Drni_Port_Conversation (9.4.7) vector listing which Portal System's Aggregator (if any) is passing each of the first 2048 Port Conversation IDs (Port Conversation ID 0 to Port Conversation ID 2047). *3P_Port_Conversation_Vector-1* is encoded in a 512-octet vector as sequence of Portal System Numbers (0 for none), indexed by Port Conversation ID, based on the Drni_Port_Conversation. The first two bits provide the Portal System Number of the Portal System that is passing Port Conversation ID 0, the second two bits provide the Portal System Number of the Portal System that is passing Port Conversation ID 1, and so on until the final two bits, which provide the Portal System Number of the Portal System that is passing Port Conversation ID 2047.

9.4.3.3.6 3P Port Conversation Vector-2 TLV

The 3P Port Conversation Vector-2 TLV structure shall be as shown in Figure 9-18 and as further described in the following field definitions:

TLV_type = 3P Port Conversation Vector-2	2
3P_Port_Conversation_Vector_2_Length = 512	
3P_Port_Conversation_Vector-2	512

Figure 9-18—3P Port Conversation Vector-2 TLV

a) *TLV_type = 3P Port Conversation Vector-2.* This field indicates the nature of the information carried in this TLV-tuple. *3P Port Conversation Vector-2* is identified by the integer value 15.
b) *3P_Port_Conversation_Vector_2_Length.* This field indicates the length (in octets) of this TLV-tuple. *3P Port Conversation Vector-2* uses a length value of 512.
c) *3P_Port_Conversation_Vector-2.* The field provides the operational Drni_Port_Conversation (9.4.7) vector listing which Portal System's Aggregator (if any) is passing each of the last 2048 Port Conversation IDs (Port Conversation ID 2048 to Port Conversation ID 4095). *3P_Port_Conversation_Vector-2* is encoded in a 512-octet vector as sequence of Portal System Numbers (0 for none), indexed by Port Conversation ID, based on the Drni_Port_Conversation. The first two bits provide the Portal System Number of the Portal System that is passing Port Conversation ID 2048, the second two bits provide the Portal System Number of the Portal System that is passing Port Conversation ID 2049, and so on until the final two bits, which provide the Portal System Number of the Portal System that is passing Port Conversation ID 4095.

9.4.3.4 Network/IPL sharing TLVs

These TLVs are only required when the Network/IPL sharing method used is one of Network / IPL sharing by tag (9.3.2.2), or Network / IPL sharing by encapsulation (9.3.2.3) in order to ensure consistent configuration among the Portal Systems. The Network / IPL sharing by time (9.3.2.1) method requires the exchange of the Network/IPL Sharing Method TLV (9.4.3.4.1) but not the Network/IPL Sharing Encapsulation TLV (9.4.3.4.2).

NOTE—No Network/IPL sharing TLVs are required when the Network/IPL sharing method used is the physical or aggregated method in 9.3.2.

Table 9-10 provides a list of the TLVs that are applicable for Network/IPL sharing methods. They are required to support the functionality described in 9.4.20.

Table 9-10—Type field values of Network/IPL sharing TLVs

TLV	Type field
Network/IPL Sharing Method TLV	16
Network/IPL Sharing Encapsulation TLV	17

9.4.3.4.1 Network/IPL Sharing Method TLV

The Network/IPL Sharing Method TLV structure shall be as shown in Figure 9-19 and as further described in the following field definitions:

TLV_type = Network/IPL Sharing Method	2
Network/IPL_Sharing_Method_Length = 4	
DRF_Home_Network/IPL_Sharing_Method	4

Figure 9-19—Network/IPL Sharing Method TLV

a) *TLV_type = Network/IPL Sharing Method TLV*. This field indicates the nature of the information carried in this TLV-tuple. The Network/IPL sharing TLVs is identified by the integer value 16.
b) *Network/IPL_Sharing_Method_Length.* This field indicates the length (in octets) of this TLV-tuple. The Network/IPL sharing TLVs uses a length value of 4.
c) *DRF_Home_Network/IPL_Sharing_Method.* This field contains the value representing the Network/IPL sharing method that is used to transport IPL frames to the Neighbor Portal System on this IPP when the IPL and network link are sharing the same physical link. It consists of the 3-octet OUI or CID identifying the organization that is responsible for this encapsulation and one following octet used to identify the encapsulation method defined by that organization. Always set equal to aDrniEncapsulationMethod (7.4.1.1.17). A value of 1 indicates that Network / IPL sharing by time (9.3.2.1) is used. A value of 2 indicates that the encapsulation method used is the same as the one used by network frames and that Network / IPL sharing by tag (9.3.2.2) is used. Table 9-11 provides the IEEE 802.1 OUI (00-80-C2) encapsulation method encodings.

Table 9-11—IEEE encapsulation methods

Encapsulation Method field	Value
IPL is using a separate physical or Aggregation link	0
Network / IPL sharing by time (9.3.2.1)	1
Network / IPL sharing by tag (9.3.2.2)	2
IEEE 802.1Q I-TAG based encapsulation	3
IEEE 802.1Q B-VLAN based encapsulation	4
IETF Pseudowire based encapsulation	5
Reserved	6–255

9.4.3.4.2 Network/IPL Sharing Encapsulation TLV

The Network/IPL Sharing Encapsulation TLV structure shall be as shown in Figure 9-20 and as further described in the following field definitions:

TLV_type = Network/IPL Sharing Encapsulation	2
Network/IPL_Sharing_Encapsulation_Length = 32	
DRF_Home_Network/IPL_IPLEncap_Digest	16
DRF_Home_Network/IPL_NetEncap_Digest	16

Figure 9-20—Network/IPL Sharing Encapsulation TLV

a) *TLV_type = Network/IPL Sharing Encapsulation TLV.* This field indicates the nature of the information carried in this TLV-tuple. The Network/IPL sharing TLVs is identified by the integer value 17.
b) *Network/IPL_Sharing_Encapsulation_Length.* This field indicates the length (in octets) of this TLV-tuple. The Network/IPL sharing TLVs uses a length value of 32.
c) *DRF_Home_Network/IPL_IPLEncap_Digest.* This field contains the value of the MD5 digest computed from aDrniIPLEncapMap (7.4.1.1.18) for exchange with the Neighbor Portal System on the IPL.
d) *DRF_Home_Network/IPL_NetEncap_Digest.* This field contains the value of the MD5 digest computed from aDrniNetEncapMap (7.4.1.1.19) for exchange on the shared network link.

9.4.3.5 Organization-Specific TLV

These optional Organization-Specific TLVs are provided to allow different organizations, such as IEEE, ITU-T, IETF, as well as individual software and equipment vendors, to define TLVs that advertise additional information to Neighbor Portal Systems. The Organization-Specific TLV structure shall be as shown in Figure 9-21 and as further described in the following field definitions:

TLV_type = Organization-Specific	2
Organization_Specific_Length = LL	
OUI/CID	3
Subtype	7
Value (Optional)	LL-10

Figure 9-21—Organization-Specific TLV

a) *TLV_type = Organization-Specific TLV.* This field indicates the nature of the information carried in this TLV-tuple. The Organization-Specific TLV is identified by the integer value 18.
b) *Organization_Specific_Length.* This field indicates the length (in octets) of this TLV-tuple. The Organization-Specific TLV uses a length value of LL.
c) *OUI/CID.* This field contains the 3-byte long OUI or CID, obtainable from the IEEE.
d) *Subtype.* This field contains a Subtype value, so that an additional OUI/CID will not be required if more Organization-Specific TLVs are required by an owner of an OUI/CID.
e) *Value.* This field contains the information that will be communicated to a Neighbor Portal System.

The following restrictions apply:

f) Information transmitted in an Organization-Specific TLV shall not require the recipient to violate any requirement in this standard;

g) Information transmitted in one Organization-Specific TLV shall not be used to provide a means for sending messages that are larger than would fit within a single DRCPDU.

9.4.4 DRCP Control Parser/Multiplexer

There can be up to two DRCP Control Parser/Multiplexers on a Portal System, one for each IPP. Each DRCP Control Parser/Multiplexer N, where N identifies one of the two IPPs, is an instance of the Protocol Parser/Multiplexer described in 6.2.7. Specifically:

a) The *DownPort* of the Protocol Parser/Multiplexer is *MacIppN* (9.3.4.1) for DRCP Control Parser/Multiplexer N.
b) The *ControlPort* of the Protocol Parser/Multiplexer is *DRCPCtrlMuxN* (9.3.4.1) for DRCP Control Parser/Multiplexer N.
c) The *DataPort* of the Protocol Parser/Multiplexer is *IppN* (9.3.4.1) for DRCP Control Parser/Multiplexer N.
d) The IsControlFrame function is defined in 9.4.4.1.

9.4.4.1 Control Parser state diagram

9.4.4.1.1 Control Parser Function

IsControlFrame
Returns Boolean value: (DA == DRCP_Protocol_DA && Length/Type == DRCP_Type && Subtype == DRCP_subtype)

9.4.4.1.2 Constants

DRCP_Type
The value of the Distributed Relay Control Protocol EtherType field.
DRCP_subtype
The value of the Subtype field for the DRCP.
Value: Integer
1

9.4.4.1.3 Variables

DA
SA
mac_service_data_unit
priority
The parameters of the M_UNITDATA.indication primitive.
DRCP_Protocol_DA
One of the addresses selected from Table 9-6 determined by the setting of the IPP Managed Objects Class managed object (7.4.2).
Value: 48 bits.
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.
Value: Integer

9.4.5 DRCP 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-IPP basis; any deviations from per-IPP description are highlighted in the text. Events (such as expiration of a timer or received DRCPDUs) may cause state transitions and also cause actions to be taken; those actions may include the need for transmission of a DRCPDU containing repeated or new information. Periodic and event-driven transmissions are controlled by the state of a Need-To-Transmit (NTTDRCPDU) variable, generated by the state machines as necessary.

The state machines are as follows:

a) *DRCPDU Receive machine (DRX—9.4.14)*. This state machine receives DRCPDUs from the Neighbor Portal System on this IPP, records the information contained, and times it out using either Short Timeouts or Long Timeouts, according to the setting of DRCP_Timeout. It evaluates the incoming information from the Neighbor Portal System to determine whether the Home and Neighbor have both agreed upon the protocol information exchanged to the extent that the Home Portal System can now be safely used, either in Portal with other Portal Systems or as an individual Portal; if not, it asserts NTTDRCPDU in order to transmit fresh protocol information to the Neighbor Portal System. If the protocol information from the Neighbor Portal System times out, the DRCPDU Receive machine installs default parameter values for use by the other state machines.
b) *DRCP Periodic Transmission machine (Periodic—9.4.15)*. This state machine determines the period that the Home Portal System and its Neighbors will exchange DRCPDUs in order to maintain the Portal.
c) *Portal System machine (PS—9.4.16)*. This state machine is responsible to update the operational status of all the Gateways and Aggregation Ports in the Portal based on local information and DRCPDUs received on the Home Portal System's IPPs. This state machine is per Portal System.
d) *DRNI Gateway and Aggregator machines (DGA—9.4.17)*. These state machines are responsible for configuring the Gateway Conversation IDs that are allowed to pass through this DR Function's Gateway and the Port Conversation IDs that are allowed to be distributed through this DR Function's Aggregator. These state machines are per Portal System.
e) *DRNI IPP machines (IPP—9.4.18)*. These state machines are responsible for configuring the Gateway Conversation IDs and the Port Conversation IDs that are allowed to pass through this DR Function's IPPs.
f) *DRCPDU Transmit machine (DTX—9.4.19)*. This state machine handles the transmission of DRCPDUs, both on demand from the other state machines, and on a periodic basis.

Figure 9-22 illustrates the relationships among these state machines and the flow of information between them. The set of arrows labeled Neighbor State Information represents new Neighbor information, contained in an incoming DRCPDU or supplied by administrative default values, being fed to each state machine by the DRCPDU Receive machine. The set of arrows labeled Home State Information represents the flow of updated Home state information between the state machines. Transmission of DRCPDUs occurs either as a result of the Periodic machine determining the need to transmit a periodic DRCPDU, or as a result of changes to the Home's state information that need to be communicated to the Neighbors. The need to transmit a DRCPDU is signaled to the DRCPDU Transmit machine by asserting NTTDRCPDU. 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.

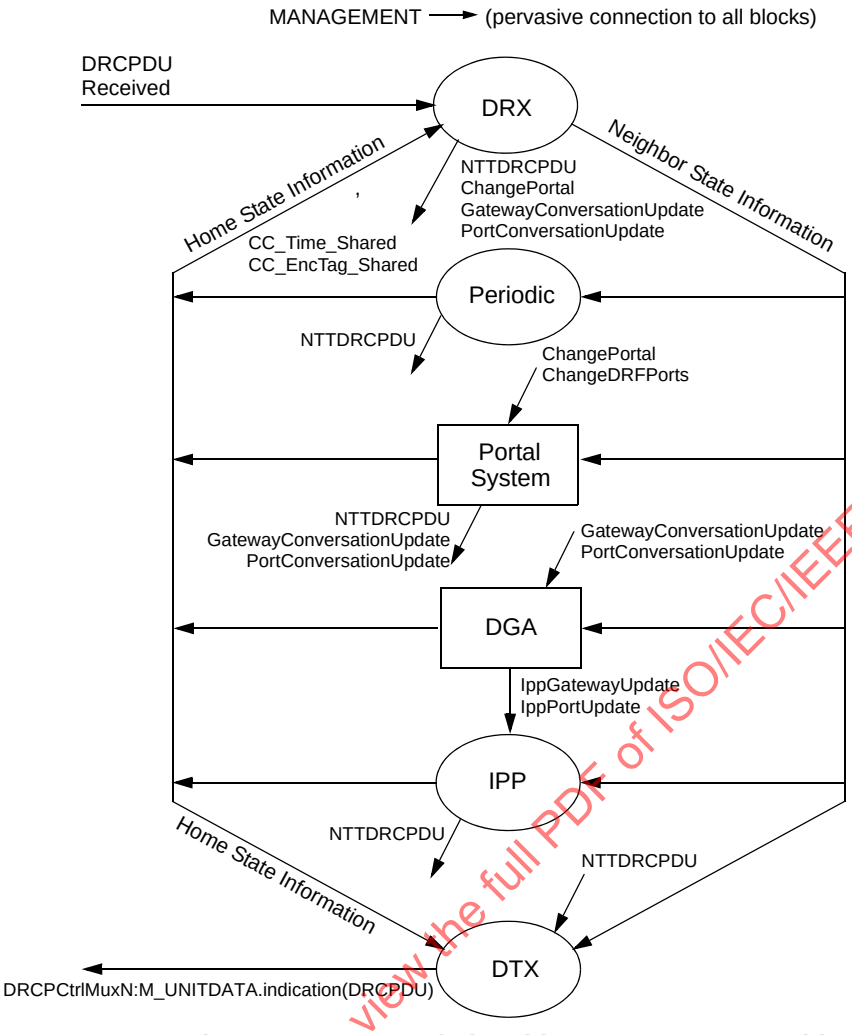


Figure 9-22—Interrelationships among state machines

9.4.6 Constants

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

Drni_Fast_Periodic_Time
: The number of seconds between periodic transmissions using Short Timeouts.
Value: Integer
1

Drni_Slow_Periodic_Time
: The number of seconds between periodic transmissions using Long Timeouts.
Value: Integer
30

Drni_Short_Timeout_Time
: The number of seconds before invalidating received DRCPDU information when using Short Timeouts (3 × Drni_Fast_Periodic_Time).
Value: Integer
3

Drni_Long_Timeout_Time

The number of seconds before invalidating received DRCPDU information when using Long Timeouts (3 × Drni_Slow_Periodic_Time).
Value: Integer
90

NOTE—Timing out DRCPDU exchanges is the method of last resort for detecting IPL failures and shifting data flows to other Portal Systems. The MAC_Operational status generated by the link hardware is the first choice for link failure detection. IEEE Std 802.1Q-2014, Clause 18, provides a method for detecting link failures (also indicated via the MAC_Operational status) when hardware means are not applicable.

9.4.7 Variables associated with the Distributed Relay

Drni_Aggregator_ID
The MAC address component of the System Identifier of the Aggregator associated with this Portal. Always set equal to aAggActorSystemID (7.3.1.1.4). Transmitted in DRCPDUs.
Value: 48 bits

Drni_Aggregator_Priority
The System Priority of the Aggregator associated to this Portal. Always set equal to aAggActorSystemPriority (7.3.1.1.5). Transmitted in DRCPDUs.
Value: Integer
Assigned by administrator or System policy.

Drni_Gateway_Conversation
Operational vector listing which Portal System's Gateway (if any) is passing each Gateway Conversation ID.
Value: sequence of Portal System Numbers (0 for none), indexed by Gateway Conversation ID.
Value computed from aDrniConvAdminGateway[] and Drni_Portal_System_State[] upon initialization and whenever the managed object or variable changes.

Drni_Port_Conversation
Operational vector listing which Portal System (if any) is passing each Port Conversation ID.
Value: sequence of Portal System Numbers (0 for none), indexed by Port Conversation ID.
Value computed from Conversation_PortList[] and Drni_Portal_System_State[] upon initialization and whenever the managed object or variable changes.

Drni_Portal_Addr
The MAC address component of the System Identifier of the Portal. Always set equal to aDrniPortalAddr (7.4.1.1.4). Transmitted in DRCPDUs.
Value: 48 bits
Assigned by administrator or System policy.

Drni_Portal_Priority
The System Priority of the Portal. Always set equal to aDrniPortalPriority (7.4.1.1.5). Transmitted in DRCPDUs.
Value: Integer
Assigned by administrator or System policy.

Drni_Three_System_Portal
A Boolean indicating if this is a Portal System that is part of a Portal consisting of three Portal Systems. Always set equal to aDrniThreePortalSystem (7.4.1.1.6). Transmitted in DRCPDUs.
Value: Boolean
Assigned by administrator or System policy.

9.4.8 Per-DR Function variables

ChangeDRFPorts
This variable tracks the operational state of the Gateway, the Gateway Vector, and all Aggregation Ports associated to this Portal System and is set to TRUE when any of them changes. This variable is also set to TRUE if new values for the Drni_Conversation_GatewayList[] or aAggConversationAdminLink[] are initiated.

Value: Boolean

ChangePortal

This variable is set to TRUE when the DRF_Neighbor_Oper_DRCP_State.IPP_Activity on any IPP on this Portal System changes. The variable can also be set to TRUE by the recordPortalConfValues function.

Value: Boolean

Drni_Common_Methods

A flag indicating whether the Gateway and the Port Algorithms use the same methods for frame distribution across the Portal Systems within a Portal. Always set equal to aDrniPortConversationControl (7.4.1.1.23). Transmitted in DRCPDUs.

Value: Boolean

Drni_Conversation_GatewayList[]

An array of 4096 lists, indexed by Gateway Conversation ID, that determines which Gateway in this Portal carries which Gateway Conversation ID. Each item in the array is a list of Gateways in this Portal, in priority order from most desired to least desired, for carrying the indexed Gateway Conversation ID. Assigned by administrator or system policy. Always set equal to aDrniConvAdminGateway[] (7.4.1.1.10).

Value: sequence of Port IDs

Drni_Portal_System_State[]

The states of all Portal Systems in this Portal, indexed by Portal System Number.

Value: For each Portal System, a Boolean flag indicating the operational state of the current Portal System's Gateway (TRUE indicates operational), the Gateway Boolean vector of the Portal System, a List (perhaps empty) of the Port IDs of the operational Aggregation Ports in that Portal System, and the identity of the IPP Intra-Portal Port, if any, from which the Portal System's state was obtained.

This variable is set by the updatePortalState function (9.4.11). Transmitted in DRCPDUs.

DRF_Home_Admin_Aggregator_Key

The administrative Aggregator Key value associated with this Portal System's Aggregator. Transmitted in DRCPDUs.

Value: Integer

Assigned by administrator or System policy. The DRF_Home_Admin_Aggregator_Key is configured and has to be different for each Portal System. Specifically the two most significant bits have to be different in each Portal System. The lower 14 bits may be any value, have to be the same in each Portal System within the same Portal, and have a default of zero.

Assigned by administrator or System policy.

DRF_Home_Conversation_GatewayList_Digest

A digest of aDrniConvAdminGateway[] (7.4.1.1.10), configured in this DR Function, for exchange with the Neighbor Portal Systems. The digest, is a 16-octet MD5 fingerprint [see IETF RFC 1321 (1992)] created from the aDrniConvAdminGateway[]. To calculate the digest, aDrniConvAdminGateway[] is considered to contain 4096 consecutive elements, where each element contains a list of Portal System Numbers, encoded as binary numbers in the order of preference, highest to lowest, followed by the Gateway Conversation ID. The first element of the table contains the prioritized list of Portal System Numbers assigned to Gateway Conversation ID 0, the second element the prioritized list of Portal System Numbers assigned to Gateway Conversation ID 1, the third element the prioritized list of Portal System Numbers assigned to Gateway Conversation ID 2, and so on, with the last element containing the prioritized list of Portal System Numbers assigned to Gateway Conversation ID 4095. This variable is referenced by the DRCPDU.

Value: MD5 digest

DRF_Home_Conversation_PortList_Digest

A digest of aAggConversationAdminLink[] (7.3.1.1.35), configured in this DR Function, for exchange with the Neighbor Portal Systems. Always set equal to Actor_Conversation_LinkList_Digest (). Transmitted in DRCPDUs.

Value: MD5 digest

DRF_Home_Gateway_Algorithm

The gateway algorithm used by this DR Function to assign frames to Gateway Conversation IDs. Always set equal to the aDrniGatewayAlgorithm (7.4.1.1.13). Transmitted in DRCPDUs.

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

DRF_Home_Gateway_Conversation_Mask

The operational Boolean Gateway vector, indexed by Gateway Conversation ID, indicating whether the indexed Gateway Conversation ID is enabled to pass through the Home Gateway (FALSE = blocked). Its value is updated by the updateDRFHomeState function and is stored as the first entry in the Gateway Vector database for the Home Portal System. Transmitted in DRCPDUs.

Value: sequence of Boolean values, indexed by Gateway Conversation ID.

DRF_Home_Gateway_Sequence

The sequence number of the operational DRF_Home_Gateway_Conversation_Mask vector. It is initialized to zero and is updated by the updateDRFHomeState function. Transmitted in DRCPDUs.

Value: Integer

DRF_Home_Port_Algorithm

The port algorithm used by this DR Function to assign frames to Port Conversation IDs. Always set equal to the associated Aggregator's aAggPortAlgorithm (7.3.1.1.33). Transmitted in DRCPDUs.

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

DRF_Home_Oper_Aggregator_Key

The operational Aggregator Key value associated with this Portal System's Aggregator. Its value is computed by the updateKey function (9.4.11). Transmitted in DRCPDUs.

Value: Integer

DRF_Home_Oper_Partner_Aggregator_Key

The operational Partner Aggregator Key associated with this Portal System's Aggregator LAG ID. Transmitted in DRCPDUs.

Value: Integer

DRF_Home_State

The operational state of this DR Function. Transmitted in DRCPDUs.

Value: Boolean flag indicating the operational state of this Portal System's Gateway (TRUE indicates operational), the Boolean Gateway Vector associated with the most recent entry in the Gateway Vector database for this Portal System, and a List (perhaps empty) of the Port IDs of the operational Aggregation Ports in this Portal System.

DRF_Neighbor_Admin_Conversation_GatewayList_Digest

The value for the digest of the prioritized Gateway Conversation ID-to-Gateway assignments of the Neighbor Portal System, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Its default value is set to NULL. It is always set equal to aDrniNeighborAdminConvGatewayListDigest (7.4.1.1.11).

Value: MD5 digest

DRF_Neighbor_Admin_Conversation_PortList_Digest

The value for the digest of the prioritized Port Conversation ID-to-Aggregation Port assignments of the Neighbor Portal System, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Its default value is set to NULL. It is always set equal to aDrniNeighborAdminConvPortListDigest (7.4.1.1.12).

Value: MD5 digest

DRF_Neighbor_Admin_DRCP_State

Default value for the Neighbor Portal System's DRCP state parameters, assigned by administrator or System policy for use when the Neighbor Portal System's information is

unknown or expired. The value consists of the following set of variables, as described in 9.4.3.2:

Home_Gateway
Neighbor_Gateway
Other_Gateway
IPP_Activity
DRCP_Timeout
Gateway_Sync
Port_Sync
Expired

Value: 8 bits

DRF_Neighbor_Admin_Gateway_Algorithm

The value for the gateway algorithm of the Neighbor Portal Systems, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Its default value is set to NULL. It is always set equal to aDrniNeighborAdminGatewayAlgorithm (7.4.1.1.14).

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

DRF_Neighbor_Admin_Port_Algorithm

The value for the port algorithm of the Neighbor Portal Systems, assigned by administrator or System policy for use when the Neighbor Portal System's information is unknown. Its default value is set to NULL. It is always set equal to aDrniNeighborAdminPortAlgorithm (7.4.1.1.15).

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

DRF_Portal_System_Number

A unique identifier for this Portal System in the Portal. Always set equal to aDrniPortalSystemNumber (7.4.1.1.7). Transmitted in DRCPDUs.

Value: An integer in the range [1...3].

PSI

This variable is set to TRUE by the updateDRFHomeState function when the Portal System is isolated from the other Portal Systems within the same Portal.

Value: Boolean

9.4.9 Per-IPP Intra-Portal Port variables

CC_Time_Shared

A Boolean indicating that Neighbor and Home Portal Systems on this IPP are consistently configured to use Network / IPL sharing by time.

Value: Boolean

CC_EncTag_Shared

A Boolean indicating that Neighbor and Home Portal Systems on this IPP are consistently configured to use Network / IPL sharing by tag or Network / IPL sharing by encapsulation as dictated by the Network / IPL method selected the aDrniEncapsulationMethod (7.4.1.1.17).

Value: Boolean

Differ_Conf_Portal

A Boolean indicating that the configured 14 least significant bit of administrative value of the key for the Portal Aggregator used by the immediate Neighbor Portal System on this IPP are different from the expected ones.

Value: Boolean

Differ_Conf_Portal_System_Number

A Boolean indicating the configured Portal System Numbers used by the immediate Neighbor Portal System on this IPP are different from the expected ones.

Value: Boolean

Differ_Gateway_Digest

A Boolean indicating that the Gateway_Digest used by the immediate Neighbor Portal System on this IPP is different from the expected one.

Value: Boolean

Differ_Port_Digest

A Boolean indicating that the Port_Digest used by the immediate Neighbor Portal System on this IPP is different from the expected one.

Value: Boolean

Differ_Portal

A Boolean indicating that the received DRCPDU on this IPP is associated with a different Portal.

Value: Boolean

DRF_Home_Conf_Neighbor_Portal_System_Number

This Portal System's configuration value for the Portal System Number of the Neighbor Portal System attached to this IPP. Always set equal to the value assigned to the two least significant bits of the priority component of the Port ID of this IPP (7.4.1.1.8). Transmitted in DRCPDUs.

Value: An integer in the range [1...3].

DRF_Home_Network/IPL_IPLEncap_Digest

A digest of aDrniIPLEncapMap (7.4.1.1.18), configured on this IPP, for exchange with the Neighbor Portal System on the IPL. The digest is calculated in a way similar to that specified for DRF_Home_Conversation_GatewayList_Digest where in place of the list of Portal System Numbers provided by aDrniConvAdminGateway[], the identifiers for the encapsulation method provided by aDrniIPLEncapMap are used instead. Transmitted in the Network/IPL Sharing Encapsulation TLV (9.4.3.4.2).

Value: MD5 digest

DRF_Home_Network/IPL_NetEncap_Digest

A digest of aDrniNetEncapMap (7.4.1.1.19), configured on this IPP, for exchange on the shared network link. The digest is calculated in a way similar to that specified for DRF_Home_Conversation_GatewayList_Digest where in place of the list of Portal System Numbers provided by aDrniConvAdminGateway[], the identifiers for he translated values of the identifiers used for a network frame provided by aDrniNetEncapMap are used instead. Transmitted in the Network/IPL Sharing Encapsulation TLV (9.4.3.4.2).

Value: MD5 digest

DRF_Home_Network/IPL_Sharing_Method

The Network/IPL sharing method used by this DR Function to share this IPP with network data. Always set equal to the aDrniEncapsulationMethod (7.4.1.1.13). Transmitted in the Network/IPL Sharing Method TLV (9.4.3.4.1) when the aDrniEncapsulationMethod is not set to the default NULL value.

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this method followed by one octet identifying this specific method).

DRF_Home_Oper_DRCP_State

The operational values of this Portal System's DRCP state parameters as reported on this IPP. This consists of the following set of variables, as described in 9.4.3.2:

- Home_Gateway
- Neighbor_Gateway
- Other_Gateway
- IPP_Activity
- DRCP_Timeout
- Gateway_Sync
- Port_Sync
- Expired

Value: 8 bits

DRF_Neighbor_Admin_Aggregator_Key

The administrative Aggregator Key value of the Neighbor Portal System on this IPP. Transmitted in DRCPDUs.
Value: Integer

DRF_Neighbor_Aggregator_ID
The last received, MAC address component of Aggregator System ID of the Neighbor Portal System, on this IPP.
Value: 48 bits

DRF_Neighbor_Aggregator_Priority
The last received, System Priority of the Neighbor Portal System's Aggregator, on this IPP.
Value: Integer

DRF_Neighbor_Conversation_GatewayList_Digest
The last-received Gateway Conversation ID digest of the Neighbor Portal System on this IPP.
Value: MD5 digest

DRF_Neighbor_Conversation_PortList_Digest
The last-received Port Conversation ID digest of the Neighbor Portal System on this IPP.
Value: MD5 digest

DRF_Neighbor_Gateway_Algorithm
The value of the algorithm used by the Neighbor Portal System to assign frames to Gateway Conversation IDs received on this IPP.
Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

DRF_Neighbor_Gateway_Conversation_Mask
The operational value of this Portal System's view of the immediate Neighbor Portal System's Gateway Vector on this IPP. This is a Boolean vector, indexed by Gateway Conversation ID, indicating whether the indexed Gateway Conversation ID is enabled to pass through the Neighbor Gateway (FALSE = blocked). Its value is initialized to a NULL vector and is updated by the recordNeighborState function to the Gateway Vector on the first entry of the Gateway Vector database for the Neighbor Portal System on this IPP.
Value: sequence of Boolean values, indexed by Gateway Conversation ID.

DRF_Neighbor_Gateway_Sequence
The sequence number of the operational DRF_Neighbor_Gateway_Conversation_Mask vector. It is initialized to zero and is updated by the recordNeighborState function. Transmitted in DRCPDUs.
Value: Integer

DRF_Neighbor_Network/IPL_IPLEncap_Digest
The last-received digest of aDrniIPLEncapMap of the Neighbor Portal System on this IPP.
Value: MD5 digest

DRF_Neighbor_Network/IPL_NetEncap_Digest
The last-received digest of aDrniNetEncapMap, for exchange on the shared network link of the Neighbor Portal System on this IPP.
Value: MD5 digest

DRF_Neighbor_Network/IPL_Sharing_Method
The last-received Network/IPL sharing method used of the Neighbor Portal System on this IPP.
Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this method followed by one octet identifying this specific method).

DRF_Neighbor_Oper_Aggregator_Key
The last-received operational Aggregator Key value of the Neighbor Portal System on this IPP.
Value: Integer

DRF_Neighbor_Oper_Partner_Aggregator_Key
The operational Partner Aggregator Key value of the Neighbor Portal System on this IPP. Transmitted in DRCPDUs.
Value: Integer

DRF_Neighbor_Oper_DRCP_State

The operational value of this Portal System's view of the current values of the Neighbor Portal System's DRCP state parameters. The Home DR Function sets this variable to the value received from the Neighbor Portal System in an DRCPDU. The value consists of the following set of variables, as described in 9.4.3.2:

Home_Gateway
Neighbor_Gateway
Other_Gateway
IPP_Activity
DRCP_Timeout
Gateway_Sync
Port_Sync
Expired

Value: 8 bits

DRF_Neighbor_Conf_Portal_System_Number

The Neighbor Portal System's configuration Portal System Number value for this Portal System that was last received on this IPP.

Value: An integer in the range [1...3].

DRF_Neighbor_Port_Algorithm

The value of the algorithm used by the Neighbor Portal System to assign frames to Port Conversation IDs received on this IPP.

Value: 4-octet (3-octet OUI or CID identifying the organization that is responsible for defining this algorithm followed by one octet identifying this specific algorithm).

DRF_Neighbor_Portal_System_Number

The last received identifier of the Neighbor Portal System on this IPP.

Value: An integer in the range [1...3].

DRF_Neighbor_State

The operational state of the immediate Neighbor Portal System on this IPP.

Value: Boolean flag indicating the operational state of the Neighbor Portal System's Gateway (TRUE indicates operational), the Boolean Gateway Vector associated with the most recent entry in the Gateway Vector database for the Neighbor Portal System, and a List (perhaps empty) of the Port IDs of the operational Aggregation Ports of the Neighbor Portal System on this IPP.

DRF_Other_Neighbor_Admin_Aggregator_Key

The administrative Aggregator Key value of the Other neighbor Portal System associated this IPP. Transmitted in DRCPDUs.

Value: Integer

DRF_Other_Neighbor_Gateway_Conversation_Mask

The operational value of this Portal System's view of the Other neighbor Portal System's Gateway Vector on this IPP. This is a Boolean vector, indexed by Gateway Conversation ID, indicating whether the indexed Gateway Conversation ID is enabled to pass through the Other neighbor Gateway (FALSE = blocked). Its value is initialized to a NULL vector and is updated by the recordNeighborState function to the Gateway Vector on the first entry of the Gateway Vector database for the Other neighbor Portal System on this IPP. Transmitted in DRCPDUs.

Value: sequence of Boolean values, indexed by Gateway Conversation ID.

DRF_Other_Neighbor_Gateway_Sequence

The sequence number of the operational DRF_Other_Neighbor_Gateway_Conversation_Mask vector. It is initialized to zero and is updated by the recordNeighborState function. Transmitted in DRCPDUs.

Value: Integer

DRF_Other_Neighbor_Oper_Partner_Aggregator_Key

The operational Partner Aggregator Key value of the Other neighbor Portal System associated this IPP. Transmitted in DRCPDUs.

Value: Integer

DRF_Other_Neighbor_State

The operational state of the Other neighbor Portal System on this IPP.
Value: Boolean flag indicating the operational state of the Other neighbor Portal System's Gateway (TRUE indicates operational), the Boolean Gateway Vector associated with the most recent entry in the Gateway Vector database for the Other neighbor Portal System, and a List (perhaps empty) of the Port IDs of the operational Aggregation Ports of the Other neighbor Portal System on this IPP.

DRF_Rcv_Home_Gateway_Conversation_Mask
The operational value of this Portal System's Gateway Vector as reported by the Neighbor Portal System on this IPP. Its value is updated by the recordNeighborState function.
Value: sequence of Boolean values, indexed by Gateway Conversation ID.

DRF_Rcv_Home_Gateway_Sequence
The operational value of this Portal System's Gateway sequence number as reported by the Neighbor Portal System on this IPP. Its value is updated by the recordNeighborState function.
Value: Integer

DRF_Rcv_Neighbor_Gateway_Conversation_Mask
The operational value of the Neighbor Portal System's Gateway Vector as reported by the Neighbor Portal System on this IPP itself. Its value is updated by the recordNeighborState function.
Value: sequence of Boolean values, indexed by Gateway Conversation ID.

DRF_Rcv_Neighbor_Gateway_Sequence
The operational value of the Neighbor Portal System's Gateway sequence number as reported by the Neighbor Portal System on this IPP itself. Its value is updated by the recordNeighborState function.
Value: Integer

DRF_Rcv_Other_Gateway_Conversation_Mask
The operational value of the Other Portal System's Gateway Vector as reported by the Neighbor Portal System on this IPP. Its value is updated by the recordNeighborState function.
Value: sequence of Boolean values, indexed by Gateway Conversation ID.

DRF_Rcv_Other_Gateway_Sequence
The operational value of the Other Portal System's Gateway sequence number as reported by the Neighbor Portal System on this IPP. Its value is updated by the recordNeighborState function.
Value: Integer

Drni_Neighbor_Common_Methods
The last received Common_Methods flag of the Neighbor Portal System on this IPP carried within the Topology State field.
Value: Boolean

Drni_Neighbor_Gateway_Conversation
The last received operational Gateway_Conversation vector of the Neighbor Portal System on this IPP carried within the received 2P Gateway Conversation Vector TLV (9.4.3.3.1) or the received pair of 3P Gateway Conversation Vector-1 TLV (9.4.3.3.2) and 3P Gateway Conversation Vector-2 TLV (9.4.3.3.3).
Value: a 1024-octet vector of a sequence of Portal System Numbers (0 for none), indexed by Gateway Conversation ID resulting from the concatenation of the 3P_Gateway_Conversation_Vector-1 and the 3P_Gateway_Conversation_Vector-2 fields from the received pair of 3P Gateway Conversation Vector-1 TLV and 3P Gateway Conversation Vector-2 TLV, if aDrniThreePortalSystem == 1, or a 512-octet vector of a sequence of Boolean values, indexed by Gateway Conversation ID copied from the 2P Gateway Conversation Vector in the received 2P Gateway Conversation Vector TLV, if aDrniThreePortalSystem == 0.

Drni_Neighbor_Port_Conversation
The last received operational Port_Conversation vector of the Neighbor Portal System on this IPP carried within the received 2P Port Conversation Vector TLV (9.4.3.3.4) or the received pair of 3P Port Conversation Vector-1 TLV (9.4.3.3.4) and 3P Port Conversation Vector-2 TLV (9.4.3.3.6).

Value: a 1024-octet vector of a sequence of Portal System Numbers (0 for none), indexed by Port Conversation ID resulting from the concatenation of the 3P_Port_Conversation_Vector-1 and the 3P_Port_Conversation_Vector-2 fields from the received pair of 3P Port Conversation Vector-1 TLV and 3P Port Conversation Vector-2 TLV, if aDrniThreePortalSystem == 1, or a 512-octet vector of a sequence of Boolean values, indexed by Port Conversation ID copied from the 2P Port Conversation Vector in the received 2P Port Conversation Vector TLV, if aDrniThreePortalSystem == 0.

Drni_Neighbor_ONN

The last received ONN flag of the Neighbor Portal System on this IPP carried within the Topology State field.

Value: Boolean

Drni_Neighbor_Portal_Addr

The last received MAC address component of Portal's System ID of the Neighbor Portal System on this IPP.

Value: 48 bits

Drni_Neighbor_Portal_Priority

The last received System Priority of the Neighbor Portal System on this IPP.

Value: Integer

Drni_Neighbor_State[]

The last received operational value of Drni_Portal_System_State[] used by the Neighbor Portal System on this IPP. This variable is updated by the recordNeighborState function.

Value: For each Portal System, the Boolean flag indicating the operational state of the current Portal System's Gateway (TRUE indicates operational), the Boolean Gateway vector of the Portal System, and a List (perhaps empty) of the Port IDs of the operational Aggregation Ports in this Portal System as reported by the Neighbor Portal System on this IPP.

Drni_Neighbor_Three_System_Portal

The last received Boolean flag indicating if the Neighbor Portal System on this IPP is a Portal System that is part of a Portal consisting of three Portal Systems.

Value: Boolean

Enabled_Time_Shared

A Boolean indicating that Neighbor and Home Portal System on this IPP are consistently configured and the Network / IPL sharing by time methods specified in 9.3.2.1 are enabled.

Value: Boolean

Enabled_EncTag_Shared

A Boolean indicating that Neighbor and Home Portal System on this IPP are consistently configured to use the tag manipulation methods of Network / IPL sharing by tag or Network / IPL sharing by encapsulation, as dictated by the Network / IPL method, selected by the aDrniEncapsulationMethod (7.4.1.1.17).

Value: Boolean

Ipp_Other_Gateway_Conversation

The operational vector listing which Portal System (if any) is passing each Gateway Conversation ID as assigned to by the immediate Neighbor Portal System on this IPP.

Value: sequence of Portal System Numbers (0 for none), indexed by Gateway Conversation ID.

Value computed from aDrniConvAdminGateway[] and Drni_Neighbor_State[] upon initialization and whenever the managed object changes or IppGatewayUpdate is TRUE.

Ipp_Other_Port_Conversation_Portal_System

The operational vector listing which Portal System (if any) is passing each Port Conversation ID as assigned to by the immediate Neighbor Portal System on this IPP.

Value: sequence of Portal System Numbers (0 for none), indexed by Port Conversation ID.

Value computed from Conversation_PortList[] and Drni_Neighbor_State[] upon initialization and whenever the managed object changes or IppPortUpdate is TRUE.

IPP_port_enabled

A variable indicating that the IPL has been established and the IPP is operable.

Value: Boolean

TRUE if the IPP is operable (MAC_Operational == TRUE).
FALSE otherwise.

NOTE—The means by which the value of the IPP_port_enabled variable is generated by the underlying MAC is implementation-dependent.

Ipp_Portal_System_State[]
The List of the states of the Portal Systems reachable through this IPP. This variable is set by the updatePortalState function.
Value: For each Portal System, Boolean flag indicating the operational state of the current Portal System's Gateway reachable through this IPP (TRUE indicates operational), the associated Boolean Gateway Vector of the Portal System indicating the Gateway Conversation IDs that is enabled by the network control protocol to pass through the Gateway (FALSE = blocked), and a List (perhaps empty) of the Port IDs of the operational Aggregation Ports in that Portal System.
In this list, the state of the immediately adjacent Portal System is the first state in the list. The list can have at most two Portal Systems' states.

Missing_Rcv_Gateway_Con_Vector
This variable indicates that Differ_Gateway_Digest is set to TRUE and no Drni_Neighbor_Gateway_Conversation can be extracted from the last received DRCPDU.
Value: Boolean

Missing_Rcv_Port_Con_Vector
This variable indicates that Differ_Port_Digest is set to TRUE and no Drni_Neighbor_Port_Conversation can be extracted from the last received DRCPDU.
Value: Boolean

NTTDRCPDU
Need To Transmit flag.
Value: Boolean
TRUE indicates that there is new protocol information that should be transmitted on this IPP, or that the Neighbor Portal System needs to be reminded of the old information.
FALSE otherwise.

ONN
Other Non Neighbor (ONN) flag. This value is updated by the updatePortalState function and is applicable only on Portals consisting of three Portal Systems. Transmitted in DRCPDUs.
Value: Boolean
TRUE indicates that the Other Ports Information TLV is not associated with an immediate Neighbor of this Portal System. FALSE (encoded as 0) indicates that the Other Ports Information TLV is from an immediate Neighbor Portal System on the other IPP on this Portal System.

9.4.10 Variables used for managing the operation of the state machines

BEGIN
This variable indicates the initialization (or reinitialization) of the DRCP 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

DRCP_Enabled
This variable indicates that the associated IPP is operating the DRCP. If the link is not a point-to-point link, the value of DRCP_Enabled shall be FALSE. Otherwise, the value of DRCP_Enabled shall be TRUE.
Value: Boolean

HomeGatewayVectorTransmit
This variable indicates to the DRCPDU Transmit machine (9.4.19) that it needs to transmit the DRF_Home_Gateway_Conversation_Mask in the Home_Gateway_Vector field of the Home

Gateway Vector TLV (9.4.3.2). There is one HomeGatewayVectorTransmit variable per IPP on this Portal System.
Value: Boolean

GatewayConversationTransmit
This variable indicates to the DRCPDU Transmit machine (9.4.19) that it needs to transmit Gateway Conversation Vector TLVs (9.4.3.3.1, 9.4.3.3.2, 9.4.3.3.3). It is set equal to TRUE when new values for the Drni_Conversation_GatewayList[] are initiated. It is set to TRUE or FALSE as specified by the recordPortalConfValues function. There is one GatewayConversationTransmit variable per IPP on this Portal System.
Value: Boolean

GatewayConversationUpdate
This variable indicates that the per Gateway Conversation ID distributions need to be updated.
Value: Boolean

IppAllGatewayUpdate
This variable is the logical OR of the IppGatewayUpdate variables for all IPPs in this Portal System.
Value: Boolean

IppAllPortUpdate
This variable is the logical OR of the IppPortUpdate variables for all IPPs in this Portal System.
Value: Boolean

IppAllUpdate
This variable is the logical OR of the IppPortUpdate and the IppGatewayUpdate variables for all IPPs in this Portal System.
Value: Boolean

IppGatewayUpdate
This variable indicates that the per Gateway Conversation ID distributions on the associated IPP need to be updated. There is one IppGatewayUpdate variable per IPP on this Portal System.
Value: Boolean

IppPortUpdate
This variable indicates that the per Port Conversation ID distributions on the associated IPP need to be updated. There is one IppPortUpdate variable per IPP on this Portal System.
Value: Boolean

OtherGatewayVectorTransmit
This variable indicates to the DRCPDU Transmit machine (9.4.19) that it needs to transmit the DRF_Home_Gateway_Conversation_Mask in the Other_Gateway_Vector field of the Other Gateway Vector TLV (9.4.3.2). There is one OtherGatewayVectorTransmit variable per IPP on this Portal System.
Value: Boolean

PortConversationTransmit
This variable indicates to the DRCPDU Transmit machine (9.4.19) that it needs to transmit Port Conversation Vector TLVs (9.4.3.3.4, 9.4.3.3.5, 9.4.3.3.6). It is set equal to TRUE when new values for the aAggConversationAdminLink[] (7.3.1.1.35) are initiate. It is set to TRUE or FALSE as specified by the recordPortalConfValues function. There is one PortConversationTransmit variable per IPP on this Portal System.
Value: Boolean

PortConversationUpdate
This variable indicates that the per Port Conversation ID distributions need to be updated.
Value: Boolean

9.4.11 Functions

InitializeDRNIGatewayConversation
: This function sets the Drni_Portal_System_Gateway_Conversation to a sequence of zeros, indexed by Gateway Conversation ID.

InitializeDRNIPortConversation
: This function sets the Drni_Portal_System_Port_Conversation to a sequence of zeros, indexed by Port Conversation ID.

InitializeIPPGatewayConversation
: This function sets the Ipp_Gateway_Conversation_Direction to a sequence of zeros, indexed by Gateway Conversation ID.

InitializeIPPPortConversation
: This function sets the Ipp_Port_Conversation_Passes to a sequence of zeros, indexed by Port Conversation ID.

recordDefaultDRCPDU
: This function sets the current Neighbor Portal System's operational parameter values to the default parameter values provided by the administrator as follows:
DRF_Neighbor_Port_Algorithm = DRF_Neighbor_Admin_Port_Algorithm;
DRF_Neighbor_Gateway_Algorithm = DRF_Neighbor_Admin_Gateway_Algorithm;
DRF_Neighbor_Conversation_PortList_Digest
= DRF_Neighbor_Admin_Conversation_PortList_Digest;
DRF_Neighbor_Conversation_GatewayList_Digest
= DRF_Neighbor_Admin_Conversation_GatewayList_Digest;
DRF_Neighbor_Oper_DRCP_State = DRF_Neighbor_Admin_DRCP_State;
DRF_Neighbor_Aggregator_Priority = aAggPortPartnerAdminSystemPriority (7.3.2.1.6);
DRF_Neighbor_Aggregator_ID = aAggPortPartnerAdminSystemID (7.3.2.1.8);
Drni_Neighbor_Portal_Priority = aAggPortPartnerAdminSystemPriority (7.3.2.1.6);
Drni_Neighbor_Portal_Addr = aAggPortPartnerAdminSystemID (7.3.2.1.8);
DRF_Neighbor_Portal_System_Number
= DRF_Home_Conf_Neighbor_Portal_System_Number, and;
DRF_Neighbor_Conf_Portal_System_Number = DRF_Portal_System_Number.

In addition for the Neighbor Portal System on the IPP:
The DRF_Neighbor_State is set to NULL (the Boolean flag for the Neighbor Portal System's Gateway is set to FALSE, the Neighbor Gateway Vector is set to NULL, and the list of the operational Aggregation Ports on the Neighbor Portal System on this IPP is emptied) and if present, the DRF_Other_Neighbor_State is also set to NULL (the Boolean flag for the Other neighbor Portal System's Gateway is set to FALSE, the Other Gateway Vector is set to NULL, and the list of the operational Aggregation Ports on the Other neighbor Portal System on this IPP is emptied). No Portal System state information is available for any Portal System on this IPP;
The DRF_Neighbor_Admin_Aggregator_Key on this IPP is set to zero;
The DRF_Other_Neighbor_Admin_Aggregator_Key on this IPP is set to zero;
The DRF_Neighbor_Oper_Partner_Aggregator_Key on this IPP is set to zero;
The DRF_Other_Neighbor_Oper_Partner_Aggregator_Key on this IPP is set to zero;
The Drni_Neighbor_Gateway_Conversation on this IPP is set;
to All_Neighbor_Conversation, which is a Portal System Number vector with all its 4096 elements set to the DRF_Home_Conf_Neighbor_Portal_System_Number, if Drni_Three_System_Portal == 1 or;
to **1**, which is a Boolean Vector with all its 4096 elements set to 1, if Drni_Three_System_Portal == 0.
The Drni_Neighbor_Port_Conversation on this IPP is set
to All_Neighbor_Conversation, if Drni_Three_System_Portal == 1 or;
to **1**, if Drni_Three_System_Portal == 0 and;

The variable ChangePortal is set to TRUE.

Finally it sets CC_Time_Shared and CC_EncTag_Shared to FALSE.

recordNeighborState

This function sets DRF_Neighbor_Oper_DRCP_State.IPP_Activity to TRUE and records the parameter values for the *Drni_Portal_System_State[]* and *DRF_Home_Oper_DRCP_State* carried in a received DRCPDU [item s) in 9.4.3.2] on the IPP, as the current parameter values for Drni_Neighbor_State[] and DRF_Neighbor_Oper_DRCP_State associated with this IPP respectively. In particular, the operational Boolean Gateway Vectors for each Portal System in the Drni_Neighbor_State[] are extracted from the received DRCPDU as follows:

For the DRF_Rcv_Neighbor_Gateway_Conversation_Mask, if the *Home_Gateway* bit in the *DRF_Home_Oper_DRCP_State* carried in the received DRCPDU is 0;

DRF_Rcv_Neighbor_Gateway_Conversation_Mask is set to NULL;

Otherwise if the *Home_Gateway_Vector* field [item al) in 9.4.3.2] is present in the received *Home Gateway Vector TLV* [item ai) in 9.4.3.2];

DRF_Rcv_Neighbor_Gateway_Conversation_Mask = *Home_Gateway_Vector*;

The tuple (*Home_Gateway_Sequence*, *Home_Gateway_Vector*) in the received *Home Gateway Vector TLV* is stored as an entry in the Gateway Vector database for the Neighbor Portal System on this IPP, indexed by the received *Home_Gateway_Sequence* in increasing sequence number order, and;

The OtherGatewayVectorTransmit on the other IPP, if it exists and is operational, is set to TRUE;

Otherwise if the *Home_Gateway_Vector* field is not present in the received *Home Gateway Vector TLV;*

The OtherGatewayVectorTransmit on the other IPP, if it exists and is operational, is set to FALSE, and;

The *Home_Gateway_Sequence* [item ak) in 9.4.3.2] is used as an index for a query in the Gateway Vector database for the Neighbor Portal System on this IPP and;

If the tuple (*Home_Gateway_Sequence*, Neighbor_Gateway_Vector) is stored as the first entry in the database, then;

DRF_Rcv_Neighbor_Gateway_Conversation_Mask = Neighbor_Gateway_Vector;

Otherwise

DRF_Rcv_Neighbor_Gateway_Conversation_Mask = **1**, where **1** is a Boolean Vector with all its 4096 elements set to 1.

NOTE—If a Gateway bit is set but no valid Gateway Vector is available then to avoid looping the recipient must assume that the neighbor Portal System has unblocked its associated gateway to all conversations

For the DRF_Rcv_Home_Gateway_Conversation_Mask, if the *Neighbor_Gateway* bit in the *DRF_Home_Oper_DRCP_State* carried in the received DRCPDU is 0;

DRF_Rcv_Home_Gateway_Conversation_Mask is set to NULL;

Otherwise;

The *Neighbor_Gateway_Sequence* [item ao) in 9.4.3.2] carried in the received *Neighbor Gateway Vector TLV* [item am) in 9.4.3.2] is used as an index for a query in the Gateway Vector database of this Portal System and;

If the tuple (*Neighbor_Gateway_Sequence*, Home_Gateway_Vector) is stored in the database, then;

DRF_Rcv_Home_Gateway_Conversation_Mask = Home_Gateway_Vector;

In addition, if that is the first entry in the database, then;

The HomeGatewayVectorTransmit on this IPP is set to FALSE;

Otherwise;

The HomeGatewayVectorTransmit on this IPP is set to TRUE, and if the *Neighbor_Gateway_Sequence* value is larger than the currently used Home_Gateway_Sequence a new entry is created in Gateway Vector database of this Portal System with the tuple values (*Neighbor_Gateway_Sequence* + 1, Home_Gateway_Vector);

Otherwise

DRF_Rcv_Home_Gateway_Conversation_Mask = **1**, where **1** is a Boolean Vector with all its 4096 elements set to 1, and;

The HomeGatewayVectorTransmit is set to TRUE.

For the DRF_Rcv_Other_Gateway_Conversation_Mask, if the *Other_Gateway* bit in the *DRF_Home_Oper_DRCP_State* carried in the received DRCPDU is 0;

DRF_Rcv_Other_Gateway_Conversation_Mask is set to NULL;

Otherwise if the *Other_Gateway_Vector* field [item as) in 9.4.3.2] is present in the received *Other Gateway Vector TLV* [item ap) in 9.4.3.2];

DRF_Rcv_Other_Gateway_Conversation_Mask = *Other_Gateway_Vector*; and

If on this IPP, Drni_Neighbor_ONN == FALSE;

The tuple (*Other_Gateway_Sequence*, *Other_Gateway_Vector)* in the received *Other Gateway Vector TLV* is stored as an entry in the Gateway Vector database for the Other neighbor Portal System on this IPP, indexed by the received *Other_Gateway_Sequence* in increasing sequence number order;

Otherwise if the *Other_Gateway_Vector* field is not present in the received *Other Gateway Vector TLV;*

The *Other_Gateway_Sequence* [item ar) in 9.4.3.2] is used as an index for a query in the Gateway Vector database for the Other neighbor Portal System on this IPP and;

If the tuple (*Other_Gateway_Sequence*, Other_Gateway_Vector) is stored in the database, then;

DRF_Rcv_Other_Gateway_Conversation_Mask = Other_Gateway_Vector;

In addition, if that is the first entry in the database, then;

The OtherGatewayVectorTransmit on this IPP is set to FALSE;

Otherwise;

The OtherGatewayVectorTransmit on this IPP is set to TRUE;

Otherwise

DRF_Rcv_Other_Gateway_Conversation_Mask = **1**, where **1** is a Boolean Vector with all its 4096 elements set to 1, and;

The OtherGatewayVectorTransmit on this IPP is set to TRUE.

It also records the following variables:

The parameter values for the *Home_Gateway* in the *DRF_Home_Oper_DRCP_State*, the Gateway Vector from the most recent entry in the Gateway Vector database for the Neighbor Portal System, and the *Active_Home_Ports* in the *Home Ports Information TLV,* carried in a received DRCPDU on the IPP, are used as the current values for the DRF_Neighbor_State on this IPP and are associated with the Portal System identified by DRF_Neighbor_Portal_System_Number;

The parameter values for the *Other_Gateway* in the DRF_Home_Oper_DRCP_State, the Gateway Vector from the most recent entry in the Gateway Vector database for the Other neighbor Portal System, and the *Other_Neighbor_Ports* in the *Other Ports Information TLV,* carried in a received DRCPDU on the IPP, are used as the current values for the DRF_Other_Neighbor_State on this IPP and are associated with the Portal System identified by the value assigned to the two most significant bits of the *DRF_Other_Neighbor_Admin_Aggregator_Key* carried within the Other Ports Information TLV in the received DRCPDU. If no Other Ports Information TLV is carried in the received DRCPDU and the Portal Topology contains three Portal Systems, the DRF_Other_Neighbor_State is set to NULL (Other_Gateway is set to FALSE, the

Other_Gateway_Vector is set to NULL, and the list of the operational Aggregation Ports on the Other neighbor Portal System on this IPP is emptied) and no Portal System state information is available on this IPP for the distant Neighbor Portal System on the IPP;
DRF_Neighbor_Admin_Aggregator_Key = *DRF_Home_Admin_Aggregator_Key*;
DRF_Neighbor_Oper_Partner_Aggregator_Key
= *DRF_Home_Oper_Partner_Aggregator_Key*;
DRF_Other_Neighbor_Admin_Aggregator_Key
= *DRF_Other_Neighbor_Admin_Aggregator_Key*, and;
DRF_Other_Neighbor_Oper_Partner_Aggregator_Key
= *DRF_Other_Neighbor_Oper_Partner_Aggregator_Key*.
Both DRF_Other_Neighbor_Admin_Aggregator_Key and DRF_Other_Neighbor_Oper_Partner_Aggregator_Key are set to NULL when the received DRCPDU does not contain the Other Ports Information TLV [item ad) in 9.4.3.2].

In addition, if Network / IPL sharing by time (9.3.2.1) is supported, the function records the parameter value for the *DRF_Home_Network/IPL_Sharing_Method* carried in the received Network/IPL Sharing Method TLV (9.4.3.4.1) as the current parameter value for the DRF_Neighbor_Network/IPL_Sharing_Method and if this is the same as the System's DRF_Home_Network/IPL_Sharing_Method, it sets CC_Time_Shared to TRUE, otherwise it sets CC_Time_Shared to FALSE.

Further, if Network / IPL sharing by tag (9.3.2.2) or Network / IPL sharing by encapsulation (9.3.2.3) is supported, the function records the Neighbor Portal System's Network/IPL sharing related parameter values carried in the received Network/IPL sharing TLVs (9.4.3.4) from an IPP, as the current operational parameter values for the immediate Neighbor Portal System on this IPP as follows:
DRF_Neighbor_Network/IPL_Sharing_Method
= *DRF_Home_Network/IPL_Sharing_Method*, carried in the received Network/IPL Sharing Method TLV (9.4.3.4.1);
DRF_Neighbor_Network/IPL_IPLEncap_Digest
= *DRF_Home_Network/IPL_IPLEncap_Digest*, carried in the received Network/IPL Sharing Encapsulation TLV (9.4.3.4.2); and
DRF_Neighbor_Network/IPL_NetEncap_Digest
= *DRF_Home_Network/IPL_NetEncap_Digest* carried in the received Network/IPL Sharing Encapsulation TLV (9.4.3.4.2).
It then compares the newly updated values of the Neighbor Portal System to this Portal System's expectations and if
DRF_Neighbor_Network/IPL_Sharing_Method
== DRF_Home_Network/IPL_Sharing_Method, and
DRF_Neighbor_Network/IPL_IPLEncap_Digest
== DRF_Home_Network/IPL_IPLEncap_Digest, and
DRF_Neighbor_Network/IPL_NetEncap_Digest
== DRF_Home_Network/IPL_NetEncap_Digest, then
    it sets CC_EncTag_Shared to TRUE;
Otherwise if one or more of the comparisons shows that the values differ,
    it sets CC_EncTag_Shared to FALSE.

It then compares the Gateway operational state and Gateway Vector for each Portal System as reported by this Portal System's Drni_Portal_System_State[] to the Gateway operational state and Gateway Vector for the same Portal System as reported by the Drni_Neighbor_State[] and if any of these differ;
    It sets GatewayConversationUpdate to TRUE;
    It sets DRF_Home_Oper_DRCP_State.Gateway_Sync to FALSE and;
    If Missing_Rcv_Gateway_Con_Vector == TRUE;

The Drni_Neighbor_Gateway_Conversation on this IPP is set;
to All_Neighbor_Conversation, which is a Portal System Number vector with all its 4096 elements set to the DRF_Home_Conf_Neighbor_Portal_System_Number, if Drni_Three_System_Portal == 1 or;
to **1**, which is a Boolean Vector with all its 4096 elements set to 1, if Drni_Three_System_Portal == 0;
Otherwise;
Drni_Neighbor_Gateway_Conversation remains unchanged;
Otherwise if they are the same and DRF_Home_Oper_DRCP_State.Gateway_Sync is FALSE;
It sets GatewayConversationUpdate to TRUE and;
DRF_Home_Oper_DRCP_State.Gateway_Sync is set to TRUE;
Otherwise if they are the same and Differ_Gateway_Digest == TRUE;
It sets GatewayConversationUpdate to TRUE;
Otherwise;
GatewayConversationUpdate remains unchanged and;
DRF_Home_Oper_DRCP_State.Gateway_Sync is set to TRUE.

It also compares the list of the Port IDs of the operational Aggregation Ports for each Portal System as reported by this Portal System's Drni_Portal_System_State[] to list of the Port IDs of the operational Aggregation Ports for the same Portal Systems as reported by the Drni_Neighbor_State[] and
if any of these differ;
It sets PortConversationUpdate to TRUE and;
DRF_Home_Oper_DRCP_State.Port_Sync is set to FALSE and;
If Missing_Rcv_Port_Con_Vector == TRUE;
The Drni_Neighbor_Port_Conversation on this IPP is set
to All_Neighbor_Conversation, if Drni_Three_System_Portal == 1 or;
to **1**, if Drni_Three_System_Portal == 0;
Otherwise;
Drni_Neighbor_Port_Conversation remains unchanged;
Otherwise if they are the same and Differ_Port_Digest == TRUE
It sets PortConversationUpdate to TRUE;
Otherwise if they are the same and DRF_Home_Oper_DRCP_State.Port_Sync is FALSE;
It sets PortConversationUpdate to TRUE and;
DRF_Home_Oper_DRCP_State.Port_Sync is set to TRUE;
Otherwise;
PortConversationUpdate remains unchanged and
DRF_Home_Oper_DRCP_State.Port_Sync is set to TRUE.

recordPortalConfValues
This function records the Neighbor Portal System's values carried in the *Portal Configuration Information TLV* of a received DRCPDU (9.4.3.2) from an IPP, as the current operational parameter values for the immediate Neighbor Portal System on this IPP as follows:
DRF_Neighbor_Portal_System_Number = *DRF_Portal_System_Number*;
DRF_Neighbor_Conf_Portal_System_Number
= *DRF_Home_Conf_Neighbor_Portal_System_Number*;
Drni_Neighbor_Three_System_Portal = *Drni_Three_System_Portal*;
Drni_Neighbor_Common_Methods = *Drni_Common_Methods*;
Drni_Neighbor_ONN = *ONN*;
DRF_Neighbor_Oper_Aggregator_Key = *DRF_Home_Oper_Aggregator_Key*;
DRF_Neighbor_Port_Algorithm = *DRF_Home_Port_Algorithm*;
DRF_Neighbor_Conversation_PortList_Digest = *DRF_Home_Conversation_PortList_Digest*;
DRF_Neighbor_Gateway_Algorithm = *DRF_Home_Gateway_Algorithm*; and

DRF_Neighbor_Conversation_GatewayList_Digest
= *DRF_Home_Conversation_GatewayList_Digest*.

It then compares the newly updated values of the Neighbor Portal System to this Portal System's expectations and if the comparison of
DRF_Neighbor_Portal_System_Number
to DRF_Home_Conf_Neighbor_Portal_System_Number, or
DRF_Neighbor_Conf_Portal_System_Number to DRF_Portal_System_Number, or
Drni_Neighbor_Three_System_Portal to Drni_Three_System_Portal, or
Drni_Neighbor_Common_Methods to Drni_Common_Methods, or
the 14 least significant bits of DRF_Neighbor_Oper_Aggregator_Key to
the 14 least significant bits of DRF_Home_Oper_Aggregator_Key, or
DRF_Neighbor_Port_Algorithm to DRF_Home_Port_Algorithm, or
DRF_Neighbor_Conversation_PortList_Digest
to DRF_Home_Conversation_PortList_Digest, or
DRF_Neighbor_Gateway_Algorithm to DRF_Home_Gateway_Algorithm, or
DRF_Neighbor_Conversation_GatewayList_Digest
to DRF_Home_Conversation_GatewayList_Digest show that one or more of the compared pairs differ;

The associated pairs of variables having the different values are stored in aIPPDebugDifferPortalReason (7.4.4.1.4); and

The reportToManagement function is invoked to report to the Management system the associated differences.

If only any of the first two pairs differ;

The variable Differ_Conf_Portal_System_Number is set to TRUE;

Otherwise;

The variable Differ_Conf_Portal_System_Number is set to FALSE.

In addition if the 14 least significant bits of DRF_Neighbor_Oper_Aggregator_Key ==
the 14 least significant bits of DRF_Home_Oper_Aggregator_Key;

the variable Differ_Conf_Portal is set to FALSE, and;

if DRF_Neighbor_Oper_Aggregator_Key is different than the DRF_Home_Oper_Aggregator_Key;

ChangePortal will be set to TRUE;

Otherwise;

The variable Differ_Conf_Portal is set to TRUE.

Further, if the variable Differ_Conf_Portal is set to FALSE and one or more of the comparisons
Drni_Neighbor_Three_System_Portal to Drni_Three_System_Portal, or
DRF_Neighbor_Gateway_Algorithm to DRF_Home_Gateway_Algorithm differ;

The Drni_Neighbor_Gateway_Conversation on this IPP is set:

to All_Neighbor_Conversation, which is a Portal System Number vector with all its 4096 elements set to the DRF_Home_Conf_Neighbor_Portal_System_Number, if Drni_Three_System_Portal == 1 or;

to **1**, which is a Boolean Vector with all its 4096 elements set to 1, if Drni_Three_System_Portal == 0;

The variable Differ_Gateway_Digest is set to TRUE;

Otherwise if the variable Differ_Conf_Portal is set to FALSE and;
Drni_Neighbor_Three_System_Portal == Drni_Three_System_Portal and;
DRF_Neighbor_Gateway_Algorithm == DRF_Home_Gateway_Algorithm and;

If DRF_Neighbor_Conversation_GatewayList_Digest
== DRF_Home_Conversation_GatewayList_Digest;

The variable Differ_Gateway_Digest is set to FALSE;

The variable GatewayConversationTransmit is set to FALSE and;

The variable Missing_Rcv_Gateway_Con_Vector is set to FALSE;
Otherwise if the comparison shows that the digests differ;
The variable Differ_Gateway_Digest is set to TRUE;
The variable GatewayConversationTransmit is set to TRUE; and
if the 2P Gateway Conversation Vector TLV (9.4.3.3.1) is present in the received DRCPDU and aDrniThreePortalSystem is FALSE ;
The Neighbor Portal System's *2P_Gateway_Conversation_Vector* carried in the *2P_Gateway_Conversation_Vector TLV*, is recorded as the current operational parameter value for the immediate Neighbor Portal System on this IPP as follows:
Drni_Neighbor_Gateway_Conversation = *2P_Gateway_Conversation_Vector* and;
The variable Missing_Rcv_Gateway_Con_Vector is set to FALSE;
Otherwise if the 3P Gateway Conversation Vector-1 TLV (9.4.3.3.2) and 3P Gateway Conversation Vector-2 TLV (9.4.3.3.3) are present in the received DRCPDU and aDrniThreePortalSystem is TRUE;
The variable *3P_Gateway_Conversation_Vector* created from the concatenation of the Neighbor Portal System's *3P_Gateway_Conversation_Vector-1* carried in the *3P Gateway Conversation Vector-1 TLV* with the Neighbor Portal System's *3P_Gateway_Conversation_Vector-2* carried in the *3P Gateway Conversation Vector-2 TLV*, is recorded as the current operational parameter value for the immediate Neighbor Portal System on this IPP as follows:
Drni_Neighbor_Gateway_Conversation = *3P_Gateway_Conversation_Vector* and;
The variable Missing_Rcv_Gateway_Con_Vector is set to FALSE;
Otherwise if no Gateway Conversation Vector TLVs (9.4.3.3.1, 9.4.3.3.2, 9.4.3.3.3) are present in the received DRCPDU, Drni_Neighbor_Common_Methods == Drni_Common_Methods == TRUE and the Port Conversation Vector TLVs (9.4.3.3.4, 9.4.3.3.4, 9.4.3.3.6) associated with the expected number of Portal Systems in the Portal are present in the received DRCPDU then;
If aDrniThreePortalSystem == FALSE,
Drni_Neighbor_Gateway_Conversation = *2P_Port_Conversation_Vector*;
Otherwise if aDrniThreePortalSystem == TRUE,
Drni_Neighbor_Gateway_Conversation = concatenation of 3P_Port_Conversation_Vector-1 with 3P_Port_Conversation_Vector-2;
and;
The variable Missing_Rcv_Gateway_Con_Vector is set to FALSE;
Otherwise;
The variable Drni_Neighbor_Gateway_Conversation remains unchanged and;
The variable Missing_Rcv_Gateway_Con_Vector is set to TRUE.
Finally, if the variable Differ_Conf_Portal is set to FALSE and one or more of the comparisons Drni_Neighbor_Three_System_Portal to Drni_Three_System_Portal, or DRF_Neighbor_Port_Algorithm to DRF_Home_Port_Algorithm differ;
The Drni_Neighbor_Port_Conversation on this IPP is set:
to All_Neighbor_Conversation, which is a Portal System Number vector with all its 4096 elements set to the DRF_Home_Conf_Neighbor_Portal_System_Number, if Drni_Three_System_Portal == 1 or;
to **1**, which is a Boolean Vector with all its 4096 elements set to 1, if Drni_Three_System_Portal == 0;
The variable Differ_Port_Digest is set to TRUE;
Otherwise if the variable Differ_Conf_Portal is set to FALSE and;
Drni_Neighbor_Three_System_Portal == Drni_Three_System_Portal and;

DRF_Neighbor_Port_Algorithm == DRF_Home_Port_Algorithm and;
If DRF_Neighbor_Conversation_PortList_Digest
== DRF_Home_Conversation_PortList_Digest;
The variable Differ_Port_Digest is set to FALSE and;
The variable PortConversationTransmit is set to FALSE;
The variable Missing_Rcv_Port_Con_Vector is set to FALSE;
Otherwise if the comparison shows that the digests differ;
The variable Differ_Port_Digest is set to TRUE;
The variable PortConversationTransmit is set to TRUE; and
if the 2P Port Conversation Vector TLV(9.4.3.3.4) is present in the received DRCPDU and aDrniThreePortalSystem is FALSE;
The Neighbor Portal System's *2P_Port_Conversation_Vector* carried in the *2P Port Conversation Vector TLV*, is recorded as the current operational parameter value for the immediate Neighbor Portal System on this IPP as follows:
Drni_Neighbor_Port_Conversation = *2P_Port_Conversation_Vector* and;
The variable Missing_Rcv_Port_Con_Vector is set to FALSE;
Otherwise if the 3P Port Conversation Vector-1 TLV (9.4.3.3.5) and 3P Port Conversation Vector-2 TLV (9.4.3.3.6) are present in the received DRCPDU and aDrniThreePortalSystem is TRUE;
The variable *3P_Port_Conversation_Vector* created from the concatenation of the Neighbor Portal System's *3P_Port_Conversation_Vector-1* carried in the *3P Port Conversation Vector-1 TLV* with the Neighbor Portal System's *3P_Port_Conversation_Vector-2* carried in the *3P Port Conversation Vector-2 TLV*, is recorded as the current operational parameter value for the immediate Neighbor Portal System on this IPP as follows:
Drni_Neighbor_Port_Conversation = *3P_Port_Conversation_Vector* and;
The variable Missing_Rcv_Port_Con_Vector is set to FALSE;
Otherwise if no Port Conversation Vector TLVs (9.4.3.3.4, 9.4.3.3.5, 9.4.3.3.6) are present in the received DRCPDU, Drni_Neighbor_Common_Methods == Drni_Common_Methods == TRUE and the Port Conversation Vector TLVs (9.4.3.3.4, 9.4.3.3.5, 9.4.3.3.6) associated with the expected number of Portal Systems in the Portal are present in the received DRCPDU then;
If aDrniThreePortalSystem == FALSE,
Drni_Neighbor_Port_Conversation = *2P_Gateway_Conversation_Vector*;
Otherwise if aDrniThreePortalSystem == TRUE,
Drni_Neighbor_Port_Conversation =
concatenation of 3P_Gateway_Conversation_Vector-1 with 3P_Gateway_Conversation_Vector-2;

and;
The variable Missing_Rcv_Port_Con_Vector is set to FALSE;
Otherwise;
The variable Drni_Neighbor_Port_Conversation remains unchanged and;
The variable Missing_Rcv_Port_Con_Vector is set to TRUE.

In all the operations above when the Drni_Neighbor_Gateway_Conversation or Drni_Neighbor_Port_Conversation are extracted from the Conversation Vector field in a received Conversation Vector TLV, the Drni_Neighbor_Gateway_Conversation or Drni_Neighbor_Port_Conversation will be set to All_Neighbor_Conversation, if Differ_Conf_Portal_System_Number == TRUE.

recordPortalValues

This function records the Neighbor's Portal parameter values carried in a received DRCPDU (9.4.3.2) from an IPP, as the current operational parameter values for the immediate Neighbor Portal System on this IPP, as follows:
DRF_Neighbor_Aggregator_Priority = *Drni_Aggregator_Priority*;
DRF_Neighbor_Aggregator_ID = *Drni_Aggregator_ID*;
Drni_Neighbor_Portal_Priority = *Drni_Portal_Priority*, and;
Drni_Neighbor_Portal_Addr = *Drni_Portal_Addr*.

It then compares the newly updated values of the Neighbor Portal System to this Portal System's expectations and if
DRF_Neighbor_Aggregator_Priority == Drni_Aggregator_Priority and
DRF_Neighbor_Aggregator_ID == Drni_Aggregator_ID and
Drni_Neighbor_Portal_Priority == Drni_Portal_Priority and
Drni_Neighbor_Portal_Addr == Drni_Portal_Addr then,
the variable Differ_Portal is set to FALSE;
Otherwise if one or more of the comparisons shows that the values differ,
the variable Differ_Portal is set to TRUE and the associated set of variables having the different values are available in aIPPDebugDifferPortalReason (7.4.4.1.4) and are reported to the Management system by invoking the reportToManagement function.

reportToManagement

This function alerts the Management system of the potential existence of a Portal System configuration error in this Portal due to the receipt of a misconfigured DRCPDU and sends to it the conflicting information from the misconfigured received DRCPDU.

setDefaultPortalSystemParameters

This function sets this Portal System's variables to administrative set values as follows:
Drni_Aggregator_Priority = aAggActorSystemPriority (7.3.1.1.5);
Drni_Aggregator_ID = aAggActorSystemID (7.3.1.1.4);
Drni_Portal_Priority = aDrniPortalPriority (7.4.1.1.5);
Drni_Portal_Addr = aDrniPortalAddr (7.4.1.1.4);
DRF_Portal_System_Number = aDrniPortalSystemNumber (7.4.1.1.7);
DRF_Home_Admin_Aggregator_Key = aAggActorAdminKey (7.3.1.1.7);
DRF_Home_Port_Algorithm = aAggPortAlgorithm (7.3.1.1.33);
DRF_Home_Gateway_Algorithm = aDrniGatewayAlgorithm (7.4.1.1.13);
DRF_Home_Conversation_PortList_Digest = the MD5 digest on aAggConversationAdminLink[] (7.3.1.1.35);
DRF_Home_Conversation_GatewayList_Digest = the MD5 digest on aDrniConvAdminGateway[] (7.4.1.1.10), and;
DRF_Home_Oper_DRCP_State = DRF_Neighbor_Admin_DRCP_State.

In addition, it sets the Drni_Portal_System_State[] as if all Gateways in the Portal are reported as FALSE, the Gateway Vectors are reported as NULL and no Aggregation Port on any Portal System is reported as operational, and each Portal System Gateway Vector database is set to contain a single entry having the Gateway Vector set to NULL and the associated Gateway Sequence initialized to 0.

setGatewayConversation

This function sets Drni_Gateway_Conversation to the values computed from aDrniConvAdminGateway[] and the current Drni_Portal_System_State[] as follows:
For every indexed Gateway Conversation ID, a Portal System Number is identified by choosing the highest priority Portal System Number in the list of Portal System Numbers provided by aDrniConvAdminGateway[] when only the Portal Systems having that Gateway Conversation ID enabled in the Gateway Vectors of the Drni_Portal_System_State[] variable, are included.

setIPPGatewayConversation

This function sets Ipp_Other_Gateway_Conversation as follows:

If Differ_Gateway_Digest == TRUE and Drni_Three_System_Portal == 1;

Ipp_Other_Gateway_Conversation = Drni_Neighbor_Gateway_Conversation;

Otherwise if Differ_Gateway_Digest == TRUE and Drni_Three_System_Portal == 0;

The Ipp_Other_Gateway_Conversation is extracted from the Drni_Neighbor_Gateway_Conversation as follows:

For every indexed Gateway Conversation ID in the Drni_Neighbor_Gateway_Conversation Boolean vector, the value zero bits are replaced with the DRF_Portal_System_Number and the value one bits are replaced with the DRF_Home_Conf_Neighbor_Portal_System_Number;

Otherwise if Differ_Gateway_Digest == FALSE;

This function sets Ipp_Other_Gateway_Conversation to the values computed from aDrniConvAdminGateway[] and the Drni_Neighbor_State[] as follows:

For every indexed Gateway Conversation ID, a Portal System Number is identified by choosing the highest priority Portal System Number in the list of Portal System Numbers provided by aDrniConvAdminGateway[] when only the Portal Systems having that Gateway Conversation ID enabled in the Gateway Vectors of the Drni_Neighbor_State[] variable, are included.

setIPPGatewayUpdate

This function sets the IppGatewayUpdate on every IPP on this Portal System to TRUE.

setIPPPortConversation

This function sets Ipp_Other_Port_Conversation_Portal_System as follows:

If Differ_Port_Digest == TRUE and Drni_Three_System_Portal == 1;

Ipp_Other_Port_Conversation_Portal_System = Drni_Neighbor_Port_Conversation;

Otherwise if Differ_Port_Digest == TRUE and Drni_Three_System_Portal == 0;

The Ipp_Other_Port_Conversation_Portal_System is extracted from the Drni_Neighbor_Port_Conversation as follows:

For every indexed Port Conversation ID in the Drni_Neighbor_Port_Conversation Boolean vector, the value zero bits are replaced with the DRF_Portal_System_Number and the value one bits are replaced with the DRF_Home_Conf_Neighbor_Portal_System_Number;

Otherwise if Differ_Port_Digest == FALSE;

This function sets Ipp_Other_Port_Conversation_Portal_System to the values computed from Conversation_PortList[] and the Drni_Neighbor_State[] as follows:

For every indexed Port Conversation ID, a Portal System Number is identified by choosing the highest priority Portal System Number in the list of Portal Systems Numbers provided by Conversation_PortList[] when only the operational Aggregation Ports, as provided by the associated Lists of the Drni_Neighbor_State[] variable, are included.

setIPPPortUpdate

This function sets the IppPortUpdate on every IPP on this Portal System to TRUE.

setPortConversation

This function sets Drni_Port_Conversation to the values computed from Conversation_PortList[] and the current Drni_Portal_System_State[] as follows:

For every indexed Port Conversation ID, a Portal System Number is identified by extracting the least significant two bits of the priority component of the highest priority Port ID (6.3.4) in the list of Port IDs provided by Conversation_PortList[] when only the operational Aggregation Ports, as provided by the associated Lists of the Drni_Portal_System_State[] variable, are included.

updateDRFHomeState

This function updates the DRF_Home_State based on the operational state of the local ports as follows:

It sets the Home Gateway bit to TRUE or FALSE based on the mechanisms that are used to identify the operational state of the local Gateway [TRUE indicates operable (i.e., the local DR Function is able to relay traffic through its Gateway Port and at least one of its other Ports—IPP(s) or Aggregator) and that connectivity through the local Gateway is enabled by the

operation of the network control protocol]. In addition the operational Boolean Gateway Vector, indexed by Gateway Conversation ID, is used to indicate which individual Gateway Conversation IDs are enabled by the network control protocol to pass through the Home Gateway (FALSE = blocked).

The operation above where the Boolean Gateway Vector is set by the network control protocol respects the requirements on distribution independence and fault isolation so that the frame distribution algorithm that is used to satisfy network requirements can be different from the algorithm used to assign frames to the Aggregation Ports of a LAG and that faults on the Aggregation Ports do not affect the attached network operation. On the other hand, the Gateway algorithm and the Port algorithm can use the same means for assigning a frame to a Conversation ID, so that the Gateway Conversation ID equals the Port Conversation ID to allow matching a gateway to the link carrying a given conversation in order to minimize traffic on the IPLs. In this case the Boolean Gateway Vector is not controlled by the network control protocol but is set instead to the Drni_Portal_System_Port_Conversation. Additionally, this function sets the Home Gateway bit TRUE if the Home Gateway Vector is not NULL and the local Gateway is operable (i.e., the local DR Function is able to relay traffic through its Gateway Port and its Aggregator Port). The profile of operation is configured by the aDrniPortConversationControl (7.4.1.1.23) managed object.

The current operational Boolean Gateway Vector along with its associated Gateway Sequence number is stored as a (Home Gateway Sequence, Home Gateway Vector) tuple in the Gateway Vector database for this Portal System. If there is any change in the Home Gateway Vector, the Home Gateway Sequence number of the current first entry in the Gateway Vector database for this Portal System (Home Gateway Sequence, Home Gateway Vector) is increased by one and a new first entry tuple (Home Gateway Sequence +1, New Home Gateway Vector) is created for the New Home Gateway Vector on top of the current one and HomeGatewayVectorTransmit is set to TRUE. When the Home_Gateway bit is set to FALSE, the New Home Gateway Vector is set to NULL (a 4096 Boolean Vector that has all its elements set to 0).

NOTE—Use of sequence numbers allows the receiver of a DRCPDU to test for a new vector without exhaustive comparison of vector values.

The list of operational Aggregation Ports is created by including only those Aggregation Port IDs for which the attached Aggregator reports them as having Actor_Oper_Port_State.Distributing == TRUE (condition that excludes the cases where the associated Aggregation Ports are either non operable (port_enabled = FALSE), in an EXPIRED state, or not in the LAG) and;
The PSI is set to TRUE if DRF_Neighbor_Oper_DRCP_State.IPP_Activity == FALSE on all IPPs on this Portal System, otherwise PSI is set to FALSE.
In addition, if PSI == TRUE and Home Gateway == FALSE then Actor_Oper_Port_State.Synchronization is set to FALSE on all Aggregation Ports on this Portal System.

The function also sets:

GatewayConversationUpdate to TRUE if the operational state of Gateway or the configured lists for Drni_Conversation_GatewayList[] has changed and sets PortConversationUpdate to TRUE if there has been any change in the list of the operational Aggregation Ports as reported by changes in the associated Actor_Oper_Port_State.Distributing variables or the configured lists for the aAggConversationAdminLink[], otherwise;

GatewayConversationUpdate and PortConversationUpdate remain unchanged.

updateIPPGatewayConversationDirection

This function computes a value for Ipp_Gateway_Conversation_Direction as follows:
For each Gateway Conversation ID, the value is TRUE if and only if:

a) the variables Drni_Gateway_Conversation and Ipp_Portal_System_State[] indicate that the target Portal System for this Gateway Conversation ID lies behind this IPP Intra-Portal Port, and
b) Drni_Gateway_Conversation and Ipp_Other_Gateway_Conversation are in agreement as to which Portal System should get this Gateway Conversation ID.

In addition, if Drni_Gateway_Conversation and Ipp_Other_Gateway_Conversation are in disagreement for any Gateway Conversation ID:

It sets DRF_Home_Oper_DRCP_State.Gateway_Sync to FALSE, and;
NTTDRCPDU to TRUE.

Otherwise:

DRF_Home_Oper_DRCP_State.Gateway_Sync and NTTDRCPDU are left unchanged.

Ipp_Gateway_Conversation_Direction is initialized to FALSE and recomputed whenever any of its contributing variables changes.

updateIPPPortConversationPasses

This function computes a value for Ipp_Port_Conversation_Passes as follows:
For each Port Conversation ID, the value is TRUE (ID passes) if and only if:

a) the variables Drni_Port_Conversation and Ipp_Portal_System_State[] indicate that the target Portal System for this Port Conversation ID lies behind this IPP Intra-Portal Port, and
b) Drni_Port_Conversation and Ipp_Other_Port_Conversation_Portal_System are in agreement as to which Portal System should get this Port Conversation ID.

In addition if Drni_Port_Conversation and Ipp_Other_Port_Conversation_Portal_System are in disagreement for any Port Conversation ID:

It sets DRF_Home_Oper_DRCP_State.Port_Sync to FALSE, and;
NTTDRCPDU to TRUE.

Otherwise:

DRF_Home_Oper_DRCP_State.Port_Sync and NTTDRCPDU are left unchanged.

Ipp_Port_Conversation_Passes is initialized to FALSE and recomputed whenever any of its contributing variables changes.

updateKey

This function updates the operational Aggregator Key, DRF_Home_Oper_Aggregator_Key, as follows:
If Partner_DWC == TRUE then:

DRF_Home_Oper_Aggregator_Key is set to the value of the least significant 14 bits of DRF_Home_Admin_Aggregator_Key, with the most significant two bits set to 01 so that the same operational Key will be used by all the Portal Systems in the Portal;

Otherwise, if PSI == TRUE and Home Gateway == FALSE then:

DRF_Home_Oper_Aggregator_Key is set to the value of the least significant 14 bits of DRF_Home_Admin_Aggregator_Key, with the most significant two bits set to 00 so that the associated links cannot be part of the LAG that is set between the Partner Portals (6.4.14.1).

Otherwise

DRF_Home_Oper_Aggregator_Key and all the operational keys of the Aggregation Ports on this system are set to the lowest numerical non zero value of the set comprising the values of the DRF_Home_Admin_Aggregator_Key, the DRF_Neighbor_Admin_Aggregator_Key and the DRF_Other_Neighbor_Admin_Aggregator_Key, on each IPP.

updateNTT

This function sets NTTDRCPDU to TRUE,
if any of:
DRF_Home_Oper_DRCP_State.Gateway_Sync, or;
DRF_Home_Oper_DRCP_State.Port_Sync, or;
DRF_Neighbor_Oper_DRCP_State.Gateway_Sync, or;
DRF_Neighbor_Oper_DRCP_State.Port_Sync;
is FALSE.

updatePortalState

On all operations associated with this function, information provided by the DRF_Other_Neighbor_State on an IPP is applicable only if Drni_Neighbor_ONN on the same IPP is FALSE.

This function updates the Drni_Portal_System_State[] as follows:
The information for this Portal System, DRF_Home_State, indexed by the Portal System Number, is included in Drni_Portal_System_State[]. For each of the other Portal Systems in the Portal:
If any of the other Portal System's state information is available from two IPPs in this Portal System, then:

- For that Portal System, only the Portal System state information provided by the DRF_Neighbor_State on the IPP having the other Portal System as a Neighbor Portal System will be included in Drni_Portal_System_State[], indexed by the Portal System Number.

Otherwise if a Portal System's state information is available only from a single IPP on this Portal System, then:

- that Portal System's state information, indexed by the associated Portal System Number will be included in the Drni_Portal_System_State[] irrespectively of whether that information is being provided by the DRF_Neighbor_State or the DRF_Other_Neighbor_State on this IPP. If information for a Portal System is available only from the DRF_Other_Neighbor_State on this IPP then ONN is set to TRUE on this IPP.

Every Portal System included in the Portal Topology for which Portal System state information is not available from any of the IPPs, has its associated Portal System state information in Drni_Portal_System_State[] set to NULL (the Gateway is set to FALSE and the list of the operational Aggregation Ports on the Portal System is emptied). For a Neighbor Portal System for which the DRF_Neighbor_State is NULL, Drni_Neighbor_State[] is set equal to Drni_Portal_System_State[].

This function updates also the Ipp_Portal_System_State[] for each IPP on this Portal System as follows:
If any other Portal System's state information is available from two IPPs, then:

- for every IPP on the Portal System, only the Portal System state information provided by the DRF_Neighbor_State on that IPP will be included in the associated Ipp_Portal_System_State[], indexed by the associated Portal System Number;

Otherwise;

- the DRF_Neighbor_State on an IPP, indexed by the associated Portal System Number, will be included as a first state in the corresponding Ipp_Portal_System_State[] and any other additional state associated with another Portal System reported on the received DRCPDU on this IPP, indexed by the associated Portal System Number, will be included as the second state in the Ipp_Portal_System_State[].

Similarly to the Drni_Portal_System_State[], every Portal System included in the Portal Topology for which Portal System state information is not available from any of the IPPs, has its associated Portal System state information Ipp_Portal_System_State[] set to NULL (the Gateway is set to FALSE, the Gateway Vector is set to NULL, and the list of the operational Aggregation Ports on the Portal System is emptied).

updatePortalSystemGatewayConversation

This function sets Drni_Portal_System_Gateway_Conversation as follows:

If Differ_Gateway_Digest == TRUE and Drni_Three_System_Portal == 0;

Drni_Portal_System_Gateway_Conversation = Drni_Neighbor_Gateway_Conversation;

Otherwise;

This function sets the Drni_Portal_System_Gateway_Conversation to the result of the logical AND operation between, the Boolean vector constructed from the Drni_Gateway_Conversation, by setting to FALSE all the indexed Gateway Conversation ID entries that are associated with other Portal Systems in the Portal, and the Boolean vectors constructed from all IPPs Ipp_Other_Gateway_Conversation, by setting to FALSE all the indexed Gateway Conversation ID entries that are associated with other Portal Systems in the Portal.

updatePortalSystemPortConversation

This function sets Drni_Portal_System_Port_Conversation as follows:

If Differ_Port_Digest == TRUE and Drni_Three_System_Portal == 0;

Drni_Portal_System_Port_Conversation = Drni_Neighbor_Port_Conversation;

Otherwise;

This function sets the Drni_Portal_System_Port_Conversation to the result of the logical AND operation between, the Boolean vector constructed from the Drni_Port_Conversation, by setting to FALSE all the indexed Port Conversation ID entries that are associated with other Portal Systems in the Portal, and the Boolean vector constructed from the Ipp_Other_Port_Conversation_Portal_System, by setting to FALSE all the indexed Port Conversation ID entries that are associated with other Portal Systems in the Portal.

9.4.12 Timers

DRCP_current_while_timer

This timer is used to detect whether received protocol information has expired. If DRF_Home_Oper_DRCP_State.DRCP_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 9.4.6).

DRCP_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 9.4.6), as specified in the Periodic Transmission state machine.

9.4.13 Messages

DRCPCtrlMuxN:M_UNITDATA.indication(DRCPDU)

This message is generated by the DRCP Control Parser/Multiplexer as a result of the reception of a DRCPDU, formatted as defined in 9.4.3.2.

9.4.14 DRCPDU Receive machine

The DRCPDU Receive machine shall implement the function specified in Figure 9-23 with its associated parameters (9.4.6 through 9.4.13). There is one DRCPDU Receive machine per IPP in a Portal System.

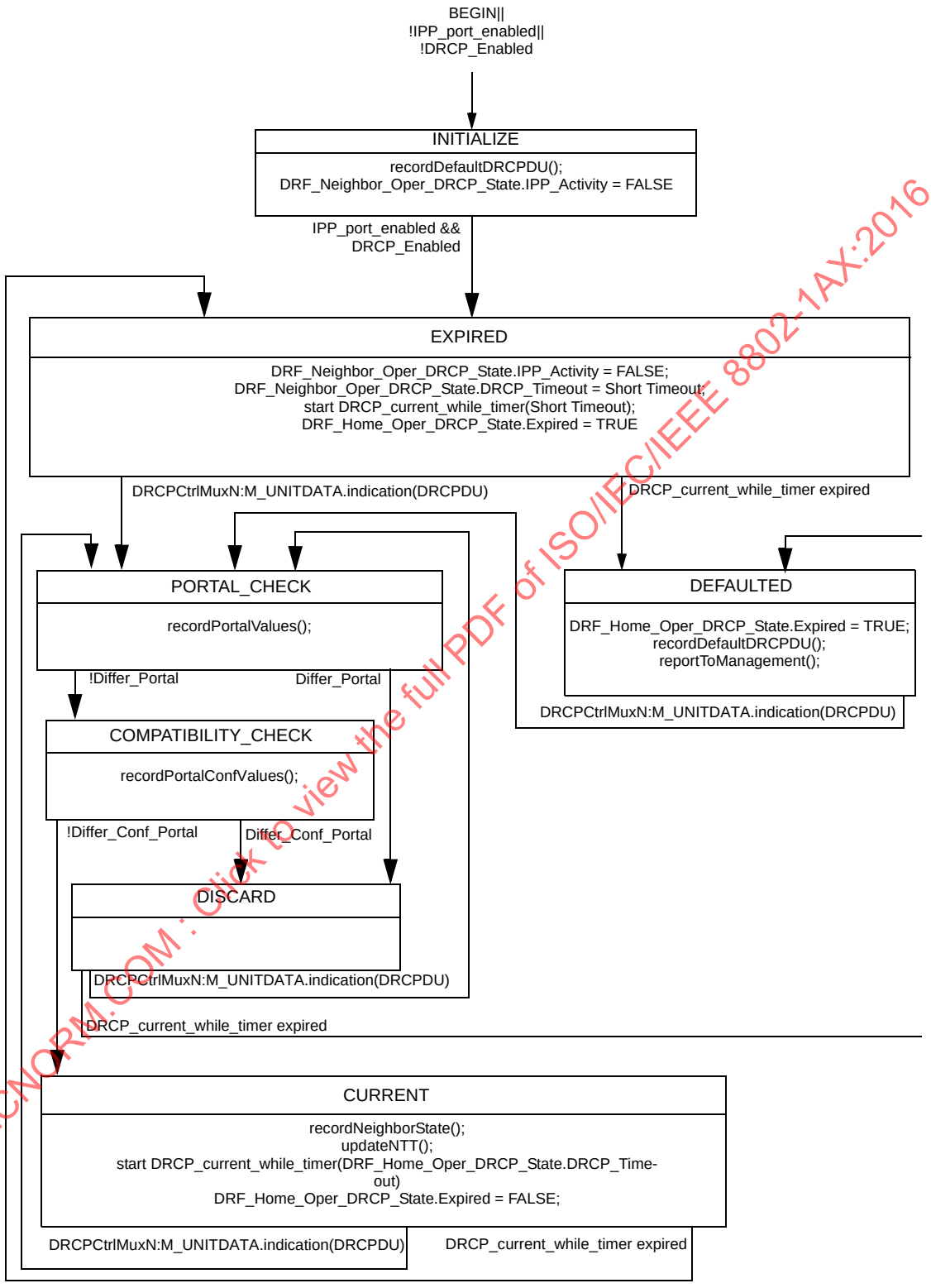


Figure 9-23—DRCPDU Receive machine state diagram

On receipt of a DRCPDU, the state machine enters the PORTAL_CHECK state. The recordPortalValues function checks if the DRCPDU is associated with this Portal. If not, the state machine will enter the REPORT_TO_MANAGEMENT state and the received DRCPDU will be reported to the management system. While in the REPORT_TO_MANAGEMENT state, the System will exit to the PORTAL_CHECK state if a new DRCPDU is received or to the EXPIRED state if the DRCP_current_while_timer expires. If the recordPortalValues identifies the received DRCPDU as associated with this Portal, it will enter the COMPATIBILITY_CHECK state to be checked by the recordPortalConfValues function. This compares the administratively configured values that are associated with this portal to the received information and if they differ the System will enter the REPORT_TO_MANAGEMENT state and the mis-configured DRCPDU will be reported to the management system. If the Portal System continues to receive DRCPDUs that do not match the administratively configured expectations for a period longer than twice the Short Timeout the state machine will transit to the DEFAULTED state and the current operational parameters for the Portal System(s) on this IPP will be overwritten with administratively configured values and a Portal System update will be triggered.

If the received DRCPDU is configured according to the expected values for this Portal the DRCPDU Receive machine enters the CURRENT state.

The recordNeighborState function records the Neighbor Portal System's State information contained in the DRCPDU in the Neighbor Portal System's State operational variables and compares it with its own (Home) Portal System state variable. If they differ, triggers are set to notify the Neighbor Portal System but also local event variables are set to trigger updates on the local Portal System machine (PS—9.4.16), the DRNI Gateway and Aggregator machines (DGA—9.4.17), and the DRNI IPP machines (IPP—9.4.18).

In the process of executing the recordPortalValues, recordPortalConfValues, and recordNeighborState, functions, a DRCPDU Receive machine compliant to this standard shall not validate the Version Number, TLV_type, or Reserved fields in received DRCPDUs. The same actions are taken regardless of the values received in these fields. A DRCPDU Receive machine may validate the Portal_Information_Length, Portal_Configuration_Information_Length, DRCP_State_Length, or Terminator_Length fields. These behaviors, together with the constraint on future protocol enhancements, are discussed in 9.4.3.2.

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

The updateNTT function is used to determine whether further protocol transmissions are required; NTTDRCPDU is set to TRUE if the Neighbor Portal System's view of the Home's operational Portal System state variable is not up to date. Then 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 Portal System's operational value of DRCP_Timeout.

If no DRCPDU is received before the DRCP_current_while_timer expires, the state machine transits to the EXPIRED state. The DRF_Neighbor_Oper_DRCP_State.IPP_Activity is set to FALSE, the current operational value of the Neighbor Portal System's DRCP_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 DRCP_Timeout settings allow the Home Portal System to transmit DRCPDUs rapidly in an attempt to reestablish communication with the Neighbor Portal System.

If no DRCPDU is received before the current_while timer expires again, the state machine transits to the DEFAULTED state. The recordDefaultDRCPDU function overwrites the current operational parameters for the Neighbor Portal System with administratively configured values and triggers further updates to reevaluate the Gateway and Port Conversation ID assignments and update its Aggregator's operational key value. The reportToManagement function is invoked in order to notify the management system for potential further actions.

If the IPP becomes inoperable, the state machine enters the INITIALIZE state. The recordDefaultDRCPDU function causes the administrative values of the Neighbor Portal System to be used as the current operational values, and the DRF_Neighbor_Oper_DRCP_State.IPP_Activity is set to FALSE. These actions force the PS machine to detach the Neighbor Portal System (and any Neighbor Portal System beyond it) from the Portal and set GatewayConversationUpdate and PortConversationUpdate to TRUE, which triggers a recomputation of the Gateway and Port Conversation ID filters.

9.4.15 DRCP Periodic Transmission machine

The DRCP Periodic Transmission machine shall implement the function specified in Figure 9-24 with its associated parameters (9.4.6 through 9.4.13). There is one DRCP Periodic Transmission machine per IPP in a Portal System.

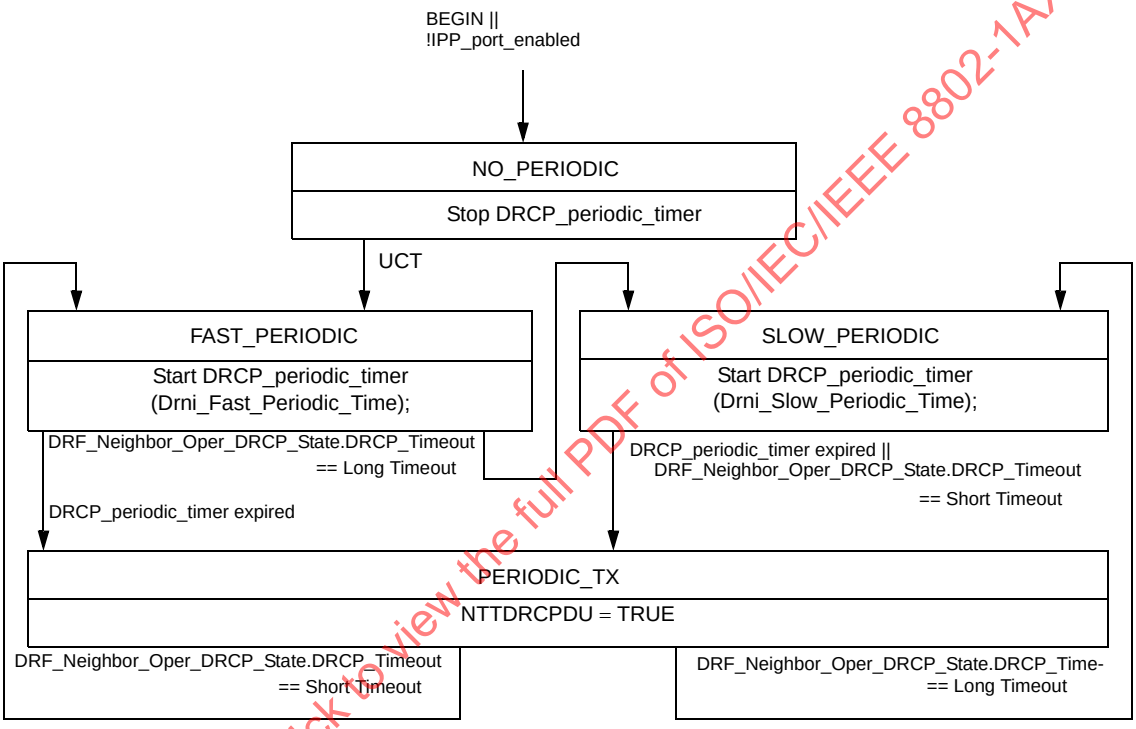


Figure 9-24—DRCP Periodic Transmission machine state diagram

The DRCP Periodic Transmission machine establishes the desire of the Home and the Neighbor Portal Systems to exchange periodic DRCPDUs on an IPP in order to maintain a Portal, 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 Neighbor Portal System; this rate is linked to the speed at which the Neighbor Portal System 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 DRCP_periodic_timer expiry, that asserts NTTDRCPDU and then exits to FAST_PERIODIC or SLOW_PERIODIC depending upon the Neighbor Portal System's DRCP_Timeout setting.

If periodic transmissions are enabled, the rate at which they take place is determined by the value of the DRF_Neighbor_Oper_DRCP_State.DRCP_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.

9.4.16 Portal System machine

The Portal System machine shall implement the function specified in Figure 9-25 with its associated parameters (9.4.6 through 9.4.13). There is one Portal System machine for each Portal System.

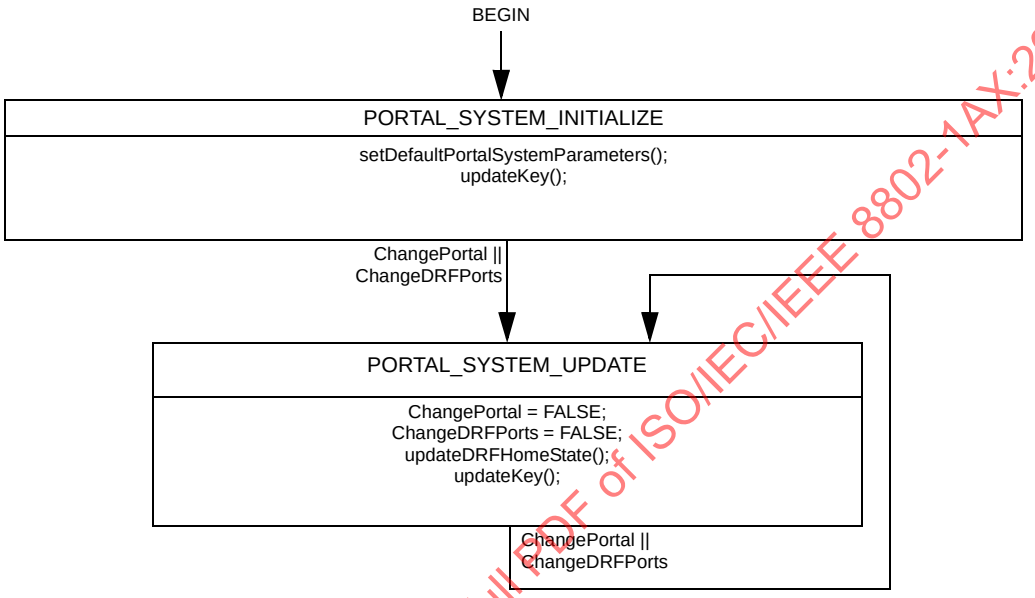


Figure 9-25—Portal System machine state diagram

The Portal System machine is responsible for updating the states of all Portal Systems in this Portal, based on information available to this Portal System.

On initialization the Portal System's variables are set to their default values for this Portal as configured by their administrative settings. In particular the default operational states of all Gateways, Aggregation Ports, and IPPs, in the Portal are set to FALSE. In addition based on those default values, the Operational Key to be used by the associated Aggregator is calculated to be the Administrative Key value assigned to this Portal System.

Any local change on the operational state of the Portal System's Gateway, or to the distribution state of any of the attached Aggregation Ports as reported by the associated Aggregator, or any change in the operational state of the Neighbor Portal Systems, as reported by the RX state machine, triggers a transition to the PORTAL_SYSTEM_UPDATE state. This causes the function updateDRFHomeState to reevaluate the variable providing the Portal System's own state (DRF_Home_State) based on the updated local information on the operational state of the Gateway and all the Aggregation Ports on the Portal System's Aggregator. Any change in the operational state of the Portal System's Gateway is reflected to the GatewayConversationUpdate that is used to trigger state transitions in the ports' state machines [DGA (9.4.17) and IPP (9.4.18)]. Similarly, any change in the operational state of the Aggregation Ports associated with this Portal System's Aggregator Port is reflected to the PortConversationUpdate that is used to trigger state transitions in the same state machines. Finally, the updateKey function updates the operational Key, to be used by the Portal System's Aggregator, by selecting the lowest numerical non zero value of the set comprising the values of the administrative Keys of all active Portal Systems in the Portal, if Conversation-

sensitive frame collection and distribution (6.6) is not supported by the two Partner Systems, or by setting the operational Key of all Partner Systems in the Portal to same value if not.

The state machine returns to the PORTAL_SYSTEM_UPDATE state whenever the operational state of any of the DR Function's ports changes.

9.4.17 DRNI Gateway and Aggregator machines

The DRNI Gateway and Aggregator machines shall implement the function specified in Figure 9-26 with their associated parameters (9.4.6 through 9.4.13). There are two DRNI Gateway and Aggregator machines on a Portal System. Each one is associated with a Conversation ID type: there is one for Gateway Conversation IDs and one for Port Conversation IDs.

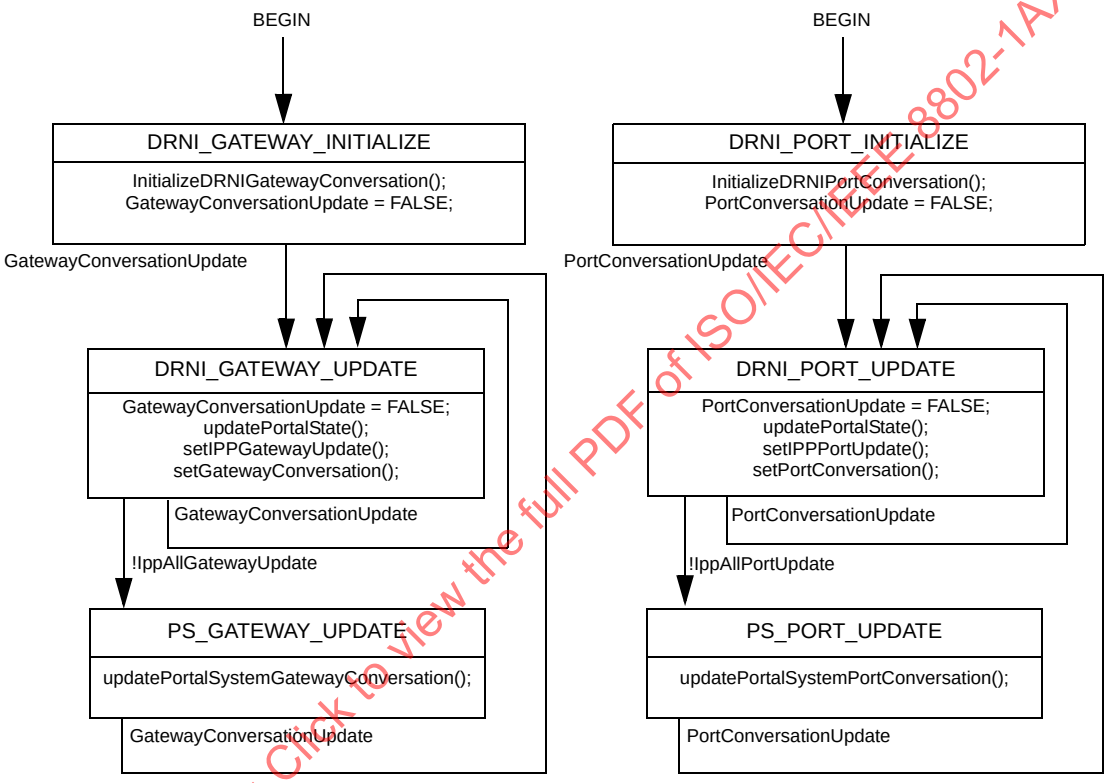


Figure 9-26—DRNI Gateway and Aggregator machines state diagrams

These state machines are responsible for configuring the Gateway Conversation IDs and the Port Conversation IDs that are allowed to pass through this DR Function's Gateway and Aggregator based on the agreed priority rules (7.3.1.1.35, 7.4.1.1.10) and the operation of DRCP.

On a trigger from the PS state machine (PS—9.4.16) or the DRX state machine (DRX—9.4.14), declaring that a Gateway's operational state has changed, the state machine enters the DRNI_GATEWAY_UPDATE state. This causes the triggering parameter (GatewayConversationUpdate) to be reset to FALSE while the function updatePortalState will update the variable providing the states of all Portal Systems (Drni_Portal_System_State[]) by combining the updated DRF_Home_State with information from the operational state of the ports on other Portal Systems as reported by the received DRCPDUs on the Portal System's IPPs and recorded by the DRX state machine (DRX—9.4.14). Subsequently, the function setIPPGatewayUpdate sets IppGatewayUpdate on every IPP on the Portal System to TRUE to trigger further updates on the IPP state machines (IPP—9.4.18) and the setGatewayConversation function is invoked to

identify the Portal System that is responsible for each Gateway Conversation ID based on the agreed selection priorities and the Gateways operational state as known by this Portal System (based on the local Gateway's operational state and the Neighbor Portal Systems' declared operational state of their own Gateways carried by the latest DRCPDU received from those Neighbor Portal Systems). Finally the, Gateway Conversation ID indexed, Boolean vector will be calculated based on the agreement between this Portal System's view on the Gateway Conversation IDs that are allowed to pass through the Portal System's Gateway and all the Neighbors' view on the Gateway Conversation IDs that are allowed to pass through this Portal System's Gateway [as declared through their DRCPDUs and recorded by this Portal System's DRX state machine (DRX—9.4.14)]. This ensures that no Gateway Conversation ID is allowed to pass through this Portal System's Gateway unless agreement between all Portal Systems is reached.

The state machine is initialized having all Gateway Conversation IDs discarded and transits to the DRNI_GATEWAY_UPDATE state whenever the trigger GatewayConversationUpdate is set.

The Port Conversation ID indexed Boolean vector is set through a similar state machine operation the only difference being the priority selection rules are based on the agreed Port Conversation IDs and the Port Algorithm instead of the agreed Gateway Conversation IDs and the Gateway Algorithm.

9.4.18 DRNI IPP machines

The DRNI IPP machines shall implement the function specified in Figure 9-27 with its associated parameters (9.4.6 through 9.4.13). There is one DRNI IPP machine per each Conversation ID type per IPP in a Portal System.

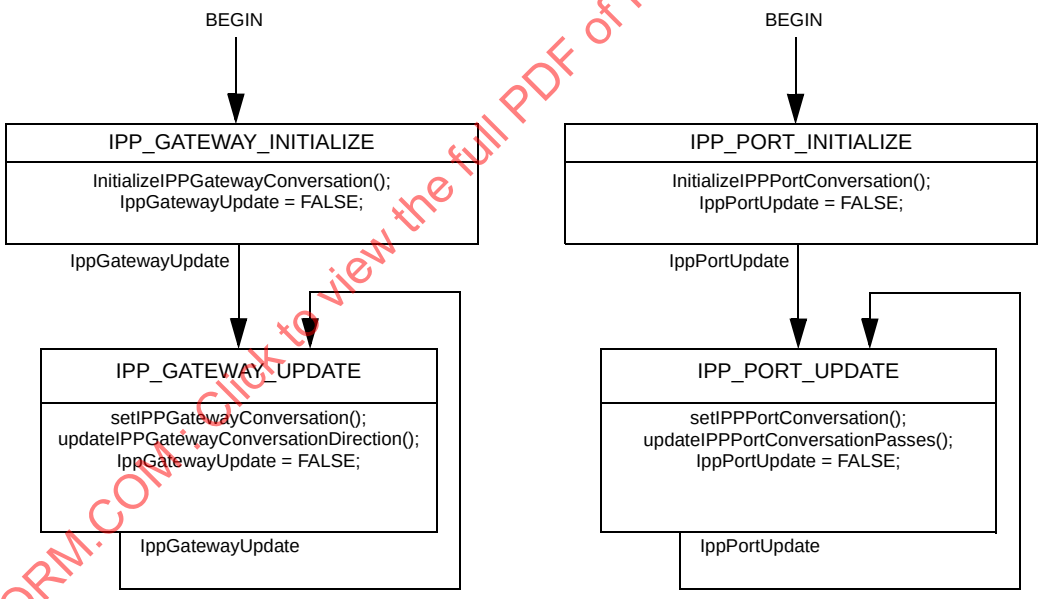


Figure 9-27—DRNI IPP machine state diagrams

These state machines are responsible for configuring the Gateway Conversation IDs and the Port Conversation IDs that are allowed to pass through this Neighbor Portal System's IPPs based on the agreed priority rules (7.3.1.1.35, 7.4.1.1.10) and the operation of DRCP.

On a trigger from the DRX state machine (DRX—9.4.14), declaring that IppGatewayUpdate is set to TRUE, the state machine enters the IPP_GATEWAY_UPDATE state. Then the setIPPGatewayConversation function will identify the Portal System that is responsible for each Gateway Conversation ID based on the agreed selection priorities and the Gateways operational states as declared by the Neighbor Portal System on this IPP (based on the Neighbor Portal System's Gateway operational state and the Neighbor Portal

System's declared operational state on their view on other Gateways in Portal, carried by the latest DRCPDU received from the Neighbor Portal System on this IPP). Subsequently, Gateway Conversation ID indexed, Boolean vector will be calculated based on the agreement between this Portal System's view on the Gateway Conversation IDs that are allowed to pass through the Portal System's IPP and the IPP Neighbor Portal System's view on the Gateway Conversation IDs that are allowed to pass through the same IPP [as declared through their DRCPDUs and recorded by this Portal System's DRX state machine (DRX—9.4.14)]. This ensures that no Gateway Conversation ID is allowed to pass through this IPP unless agreement between this Portal System and its Neighbor Portal System is reached. Finally, IppGatewayUpdate is reset to FALSE.

The state machine is initialized having all Gateway Conversation IDs discarded and transits to the IPP_GATEWAY_UPDATE state whenever the trigger GatewayConversationUpdate is set.

The Port Conversation ID indexed Boolean vector is set through a similar state machine operation, the only difference being that the priority selection rules are based on the agreed Port Conversation IDs and the Port Algorithm, instead of the agreed Gateway Conversation IDs and the Gateway Algorithm.

9.4.19 DRCPDU Transmit machine

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

a) Aggregator ID and Priority
b) Portal Address and Priority
c) Portal System Number
d) Topology State
e) Operational Aggregator Key
f) Port algorithm
g) Gateway algorithm
h) Port Digest
i) Gateway Digest
j) If GatewayConversationTransmit is TRUE and if;
    Drni_Three_System_Portal == 0;
        The 2P Gateway Conversation Vector TLV is prepared for DRCPDU transmission;
    Otherwise if Drni_Three_System_Portal == 1;
        The pair of 3P Gateway Conversation Vector-1 TLV and 3P Gateway Conversation Vector-2 TLV are prepared for DRCPDU transmission;
  If PortConversationTransmit is TRUE and if;
    Drni_Three_System_Portal == 0;
        The 2P Port Conversation Vector TLV is prepared for DRCPDU transmission;
    Otherwise if Drni_Three_System_Portal == 1;
        The 3P Port Conversation Vector-1 TLV accompanied by the 3P Port Conversation Vector-2 TLVare prepared for DRCPDU transmission;
  If both GatewayConversationTransmit and PortConversationTransmit are TRUE and if;
    Drni_Common_Methods is TRUE then;
        either of the 2P Gateway Conversation TLVs (the 2P Gateway Conversation Vector TLV, when Drni_Three_System_Portal == 0, or the pair of 3P Gateway Conversation Vector-1 TLV and 3P Gateway Conversation Vector-2 TLV, when Drni_Three_System_Portal == 1) or the 2P Port Conversation Vector TLVs (the 2P Port Conversation Vector TLV, when Drni_Three_System_Portal == 0, or the pair of 3P Port Conversation Vector-1 TLV and 3P Port Conversation Vector-2 TLV, when Drni_Three_System_Portal == 1) are sufficient to be prepared for DRCPDU transmission as each of the associated Conversation Vector TLVs set is applicable to both Gateway and Port distributions;
    Otherwise if Drni_Common_Methods is FALSE and Drni_Three_System_Portal == 1;

Two separate DRCPDUs, one including the pair of 3P Gateway Conversation Vector-1 TLV and 3P Gateway Conversation Vector-2 TLV and another one including the pair of 3P Port Conversation Vector-1 TLV and 3P Port Conversation Vector-2 TLV will be prepared for transmission having all other TLVs the same.

If present, the Conversation Vector TLVs should be inserted between the Conversation Vector Indicator TLV and the DRCP State TLV.

k) DRCP State.

l) The operational Aggregation Ports, the Administrative Aggregator Key and the operational Partner Aggregator Key of the Home Portal System and any other Portal System in the Drni_Portal_System_State[] (9.4.8).

m) The operational Gateway Sequence numbers for every Portal System in the Drni_Portal_System_State[] and, if HomeGatewayVectorTransmit and/or OtherGatewayVectorTransmit are set to TRUE and the potential presence of the Conversation Vector TLVs does not result in a DRCPDU that has a length that is larger than the maximum allowed by the access method supporting that IPP, the associated Home Gateway Vector and/or Other Gateway Vector respectively.

In addition if the system is configured to use one of the Network/IPL shared methods specified in 9.3.2.1, 9.3.2.2, or 9.3.2.3 by configuring a non-NULL value in aDrniEncapsulationMethod additional Network/IPL sharing TLVs will need to be attached to the main DRCPDU, carrying the appropriate operational values as specified in 9.4.3.4.1 and 9.4.3.4.2

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

— Not transmit any DRCPDUs, and
— Set the value of NTTDRCPDU to FALSE.

When the DRCP_Enabled variable is TRUE and the NTTDRCPDU (9.4.9) variable is TRUE, the Transmit machine shall ensure that a properly formatted DRCPDU (9.4.3.2) is transmitted [i.e., issue a DRCPCtrlMuxN:M_UNITDATA.Request(DRCPDU) service primitive]. The NTTDRCPDU variable shall be set to FALSE when the Transmit machine has transmitted a DRCPDU containing all the required fields as specified in this clause.

When the DRCP_Enabled variable is FALSE, the Transmit machine shall not transmit any DRCPDUs and shall set the value of NTTDRCPDU to FALSE.

9.4.20 Network/IPL sharing machine

The Network/IPL sharing machine shall implement the functions specified in Figure 9-28 with its associated parameters (9.4.6 through 9.4.13). There is one Network/IPL sharing machine per IPP in a Portal System for the supported Network/IPL sharing method. This machine is only required when the Network/IPL shared methods, Network / IPL sharing by time (9.3.2.1), Network / IPL sharing by tag (9.3.2.2), or Network / IPL sharing by encapsulation (9.3.2.3) is implemented.

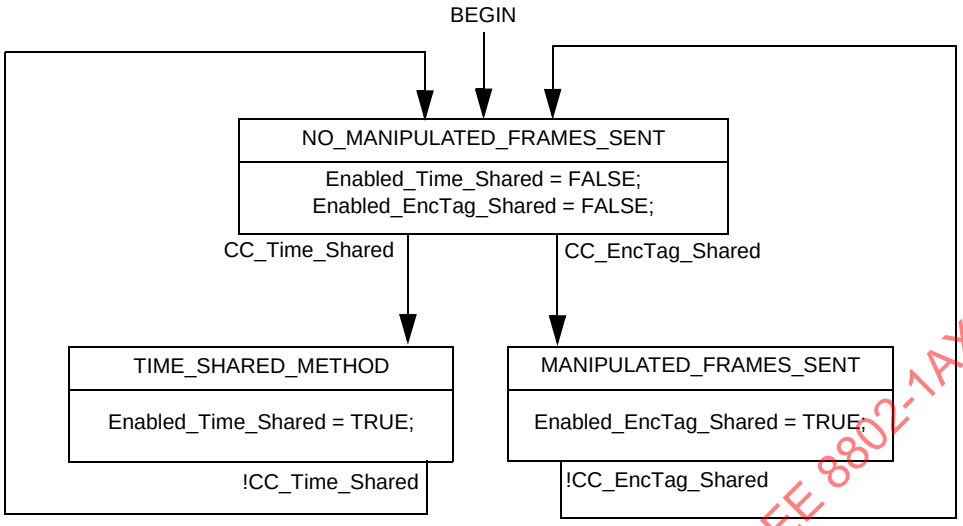


Figure 9-28—Network/IPL sharing machine state diagram

The Network/IPL sharing machine enables transmission and manipulation of the frames sent on the shared Network/IPL link only if the DRCPDUs received on the same port report the same Network/IPL sharing configuration by the Neighbor Portal System.

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

a) NO_MANIPULATED_FRAMES_SENT. While in this state, the IPL can only be supported by a physical or Aggregation Link.
b) TIME_SHARED_METHOD. While in this state, the Network / IPL sharing by time methods specified in 9.3.2.1 are enabled.
c) MANIPULATED_FRAMES_SENT. While in this state, the tag manipulation methods of Network / IPL sharing by tag or Network / IPL sharing by encapsulation, as dictated by the Network / IPL sharing method selected the aDrniEncapsulationMethod (7.4.1.1.17), are enabled.

The System is initialized in the NO_MANIPULATED_FRAMES_SENT and IPL frames are sent on the dedicated physical link. If the Home Portal System is configured for Network / IPL sharing by time mode of operation, indicated by a value of 1 in aDrniEncapsulationMethod (7.4.1.1.17), the system will transit to TIME_SHARED_METHOD if the DRX state machine (DRX—9.4.14) sets CC_Time_Shared to TRUE (indicating that the Neighbor Portal System on this IPP has also been configured for the Network / IPL sharing by time mode of operation). The System remains in the TIME_SHARED_METHOD state until a received DRCPDU sets CC_Time_Shared to FALSE, which triggers a state transition to the NO_MANIPULATED_FRAMES_SENT state and IPL frames are sent on the dedicated physical link.

Similarly, if the Home Portal System is configured for Network / IPL sharing by tag or Network / IPL sharing by encapsulation mode of operation, as indicated by the value in aDrniEncapsulationMethod (7.4.1.1.17), the system will transit to MANIPULATED_FRAMES_SENT if the DRX state machine (DRX—9.4.14) sets CC_EncTag_Shared to TRUE (indicating that the Neighbor Portal System on this IPP has also been configured for the Network / IPL sharing by tag or Network / IPL sharing by encapsulation mode of operation respectively). The System remains in the MANIPULATED_FRAMES_SENT state until a received DRCPDU sets CC_EncTag_Shared to FALSE, which triggers a state transition to the NO_MANIPULATED_FRAMES_SENT state and IPL frames are sent on the dedicated physical link.

Annex A

(normative)

Protocol Implementation Conformance Statement (PICS) proforma

A.1 Introduction[9]

The supplier of an implementation that is claimed to conform to Clause 5 shall complete the following 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.

A.1.1 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 shall be met if both optional items are implemented

[9]*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.

A.1.2 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.)

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.

A.1.3 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.

A.1.4 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.

A.1.5 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.

A.1.6 Identification

A.1.6.1 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).	

A.1.6.2 Protocol summary

Identification of protocol specification	**IEEE Std 802.1AX-2014, Link Aggregation**
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	

A.2 PICS proforma for Clause 6

A.2.1 Major capabilities/options

Item	Feature	Subclause	Value/Comment	Status	Support
	The items below are relevant to the ports of a System for which support is claimed				
LA	Is Link Aggregation supported as specified in Clause 6	Item a) in 5.3, 6.2	Support the Link Aggregation Sublayer and conform to the state machines and procedures in 6.2	M	Yes [ ]
LACP	Does the Link Aggregation Control Protocol	Item b) in 5.3, 6.3, 6.4	Support the Link Aggregation Control Protocol and conform to the state machines and procedures in 6.3 and 6.4	M	Yes [ ]
V1LA	Are Version 1 LACPDUs supported?	Item c) in 5.3, 6.4.2	Transmit and receive version 1 LACPDUs in the formats specified in 6.4.2	M	Yes [ ]
V2LA	Are Version 2 LACPDUs supported	Item h) in 5.3.1, 6.4.2	Transmit and receive version 2 LACPDUs in the formats specified in 6.4.2	O CSCD3: M	Yes [ ] No [ ]
MG	Is the Marker Generator/ Receiver supported?	Item a) in 5.3.1, 6.2.5		O	Yes [ ] No [ ]
MGT	Is Management supported?	Item b) in 5.3.1, Clause 7	Support the management functionality for Link Aggregation as specified in Clause 7	O	Yes [ ] No [ ]
MIB	Does the implementation support management operations using SMIv2 MIB modules?	Item c) in 5.3.1, Annex D	Support SMIv2 MIB modules for the management of Link Aggregation capabilities	MGT: O	N/A [ ] Yes [ ] No [ ]
AM	Is there Aggregation Port Debug Information package support?	Item d) in 5.3.1, 7.3		MGT: O	N/A [ ] Yes [ ] No [ ]
CM	Does the implementation support the Churn Detection machine?	Item d) in 5.3.1, 6.4.17	Required if Aggregation Port Debug Information package supported	AM: M !AM: O	N/A [ ] Yes [ ] No [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
PSFD	Is Per-service frame distribution supported?	Item g) in 5.3.1, 8.2		O	Yes [ ] No [ ]
CSCD	Is Conversation-sensitive frame collection and distribution supported?	Item g) in 5.3.1, A.2.27		PSFD: M	N/A [ ] Yes [ ]
DRNI	Is DRNI supported?	5.4, A.2.30		O	Yes [ ] No [ ]

A.2.2 LLDP Port connectivity

Item	Feature	Subclause	Value/Comment	Status	Support
LLDP	Does the System support LLDP?	Item e) in 5.3.1, 6.1.3		O	Yes [ ] No [ ]
LLDPPM	Does the System support LLDP Parser/Multiplexers on the Aggregation Ports?	Item e) in 5.3.1, 6.1.3		LLDP: M	N/A [ ] Yes [ ]

A.2.3 Protocol Parser/Multiplexer support

Item	Feature	Subclause	Value/Comment	Status	Support
PPM	Does the System support Protocol Parser/Multiplexer for a protocol that is supported by the System but not specified in this standard?	Item f) in 5.3.1, 6.2.7		O	N/A [] Yes [ ]

A.2.4 Frame Collector

Item	Feature	Subclause	Value/Comment	Status	Support
FC1	Frame Collector function	6.2.3	As specified in the state machine shown in Figure 6-3 and associated definitions in 6.2.3.1	M	Yes [ ]
FC2	Frame Collector function—CollectorMaxDelay	6.2.3.1.4	Deliver or discard frames within CollectorMaxDelay	M	Yes [ ]

A.2.5 Frame Distributor

Item	Feature	Subclause	Value/Comment	Status	Support
	Distribution algorithm ensures the following, when frames received by Frame Collector:	6.2.4			
FD1	Frame misordering		None	M	Yes [ ]
FD2	Frame duplication		None	M	Yes [ ]
FD3	Frame Distributor function	6.2.4	Function as specified in the state machine shown in Figure 6-4 and associated definitions in 6.2.4.1	M	Yes [ ]

A.2.6 Marker protocol

Item	Feature	Subclause	Value/Comment	Status	Support
MGR1	Marker Generator/Receiver	6.2.5		MG:M	N/A [ ] Yes [ ]
MGR2	Marker Responder	Item d) in 5.3, 6.2.6	Function specified in 6.5.4.2	!DRNI1:M DRNI1:O	N/A [ ] Yes [ ] No [ ]

A.2.7 Aggregator Parser/Multiplexer

Item	Feature	Subclause	Value/Comment	Status	Support
APM1	Aggregator Multiplexer	6.2.8	Transparent pass-through of frames	M	Yes [ ]
APM2	Aggregator Multiplexer	6.2.8	Discard of TX frames when Aggregation Port not Distributing	M	Yes [ ]
APM3	Aggregator Parser	6.2.8	Function specified by state machine shown in Figure 6-6 and associated definitions in 6.2.8.1	M	Yes [ ]
APM4	Aggregator Parser	6.2.8	Discard of RX frames when Aggregation Port not Collecting	M	Yes [ ]

A.2.8 Control Parser/Multiplexer

Item	Feature	Subclause	Value/Comment	Status	Support
CPM1	Control Multiplexer	6.2.10	Transparent pass-through of frames	M	Yes [ ]
CPM2	Control Parser	6.2.10	Function specified by state machine shown in Figure 6-5 and associated definitions in 6.2.10.1 and 6.2.9	M	Yes [ ]

A.2.9 System identification

Item	Feature	Subclause	Value/Comment	Status	Support
SID1	Globally unique identifier	6.3.2	Globally administered individual MAC address plus System Priority	M	Yes [ ]
SID2	MAC address chosen	6.3.2	MAC address associated with one of the Aggregation Ports	O	Yes [ ] No [ ]

A.2.10 Aggregator identification

Item	Feature	Subclause	Value/Comment	Status	Support
AID1	Globally unique identifier	6.3.3	Globally administered individual MAC address	M	Yes [ ]
AID2	Integer identifier	6.3.3	Uniquely identifies the Aggregator within the System	M	Yes [ ]
*AID3	Unique identifier allocated	6.3.3	Unique identifier assigned to one of its bound Aggregation Ports	O	Yes [ ] No [ ]
AID4			Unique identifier not assigned to any other Aggregator	!AID3 :M	N/A [ ] Yes [ ]

A.2.11 Port identification

Item	Feature	Subclause	Value/Comment	Status	Support
PID1	Port Identifiers	6.3.4	Unique within a System; Port Number 0 not used for any Aggregation Port	M	Yes [ ]

A.2.12 Capability identification

Item	Feature	Subclause	Value/Comment	Status	Support
CID1	Administrative and operational Key values associated with each Aggregation Port	6.3.5		M	Yes [ ]
CID2	Administrative and operational Key values associated with each Aggregator	6.3.5		M	Yes [ ]

A.2.13 Link Aggregation Group identification

Item	Feature	Subclause	Value/Comment	Status	Support
LAG1	LAG ID component values	6.3.6.1	Actor's values non-zero. Partner's admin values only zero for Individual Aggregation Ports	M	Yes [ ]

A.2.14 Detaching a link from an Aggregator

Item	Feature	Subclause	Value/Comment	Status	Support
DLA1	Effect on conversation reallocated to a different link	6.3.14	Frame ordering preserved	M	Yes [ ]

A.2.15 LACPDU structure

Item	Feature	Subclause	Value/Comment	Status	Support
LPS2	LACPDU structure	6.4.2.3	As shown in Figure 6-7 and as described	M	Yes [ ]
LPS3	LACPDU structure	6.4.2	All Reserved octets ignored on receipt and transmitted as zero	M	Yes [ ]

A.2.16 Version 2 LACPDU

Item	Feature	Subclause	Value/Comment	Status	Support
	If V2LA is not supported, mark N/A and ignore the remainder of this table				N/A [ ]
V2LA1	Is the Long LACPDU machine supported?	6.4.18	As shown in Figure 6-25 and as described	V2LA:M	Yes [ ]
V2LA2	Is the Port Algorithm TLV supported?	6.4.2.4.1	As shown in Figure 6-9 and as described	V2LA:M	Yes [ ]
V2LA3	Is the Port Conversation ID Digest TLV supported?	6.4.2.4.2	As shown in Figure 6-10 and as described	V2LA:M	Yes [ ]
V2LA4	Is the Port Conversation Mask TLVs supported?	6.4.2.4.3	As shown in Figure 6-11, Figure 6-12, Figure 6-13, Figure 6-14 and as described	V2LA:M	Yes [ ]
V2LA5	Is the Port Conversation Service Mapping TLV supported?	6.4.2.4.4	As shown in Figure 6-16 and as described	V2LA:O V2LA AND PSFD2: M	Yes [ ] No [ ]

A.2.17 State machine variables

Item	Feature	Subclause	Value/Comment	Status	Support
SMV1	Partner_Admin_Port_State	6.4.7	Collecting set to the same value as Synchronization	M	Yes [ ]
SMV2	LACP_Enabled	6.4.8	TRUE for point-to-point links, otherwise FALSE	M	Yes [ ]

A.2.18 Receive machine

Item	Feature	Subclause	Value/Comment	Status	Support
RM1	Receive machine	6.4.12	As defined in Figure 6-18 and associated parameters	M	Yes [ ]
RM2	Validation of LACPDUs		No validation of Version Number, TLV_type, or Reserved fields	M	Yes [ ]

A.2.19 Periodic Transmission machine

Item	Feature	Subclause	Value/Comment	Status	Support
PM1	Periodic Transmission machine	6.4.13	As defined in Figure 6-19 and associated parameters	M	Yes [ ]

A.2.20 Selection Logic

Item	Feature	Subclause	Value/Comment	Status	Support
	Selection logic requirements	6.4.14.1			
SLM1	Aggregator support		At least one Aggregator per System	M	Yes [ ]
SLM2	Aggregation Port Keys		Each Aggregation 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 Aggregation Port	M	Yes [ ]
SLM6	Aggregation Ports that are members of the same LAG		Aggregation Ports select same Aggregator	M	Yes [ ]
SLM7	Pair of Aggregation Ports connected in loopback		Not select same Aggregator as each other	M	Yes [ ]
SLM8	Aggregation Port required to be Individual		Not select same Aggregator as any other Aggregation Port	M	Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
SLM9	Aggregation Port is Aggregateable		Not select same Aggregator as any Individual Aggregation Port	M	Yes [ ]
SLM10	Aggregation Port unable to select an Aggregator		Aggregation Port not attached to any Aggregator	M	Yes [ ]
SLM11	Further aggregation constraints		Aggregation 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	6.4.14.2	Meets requirements of 6.4.14.2	O	Yes [ ] No [ ]

A.2.21 Mux machine

Item	Feature	Subclause	Value/Comment	Status	Support
XM1	Mux machine	6.4.15	As defined in Figure 6-21 or Figure 6-22, and associated parameters	M	Yes [ ]

A.2.22 Transmit machine

Item	Feature	Subclause	Value/Comment	Status	Support
	Transmitted in outgoing LACPDUs	6.4.16			
TM1	Actor_Port and Actor_Port_Priority	6.4.16		M	Yes [ ]
TM2	Actor_System and Actor_System_Priority	6.4.16		M	Yes [ ]
TM3	Actor_Key	6.4.16		M	Yes [ ]
TM4	Actor_State	6.4.16		M	Yes [ ]
TM5	Partner_Port and Partner_Port_Priority	6.4.16		M	Yes [ ]
TM6	Partner_System and Partner_System_Priority	6.4.16		M	Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
TM7	Partner_Key	6.4.16		M	Yes [ ]
TM8	Partner_State	6.4.16		M	Yes [ ]
TM9	CollectorMaxDelay	6.4.16		M	Yes [ ]
TM10	Action when Periodic machine is in the NO_PERIODIC state	6.4.16	Set NTT to FALSE, do not transmit	M	Yes [ ]
TM11	Action when LACP_Enabled is TRUE, NTT is TRUE, and not rate limited	6.4.16	Properly formatted LACPDU transmitted	M	Yes [ ]
TM12	Action when LACP_Enabled is TRUE and NTT is TRUE, when rate limit is in force	6.4.16	Transmission delayed until limit is no longer in force	M	Yes [ ]
TM13	Action when LACPDU has been transmitted	6.4.16	Set NTT to FALSE	M	Yes [ ]
TM14	Action when LACP_Enabled is FALSE	6.4.16	Set NTT to FALSE, do not transmit	M	Yes [ ]

A.2.23 Churn Detection machines

Item	Feature	Subclause	Value/Comment	Status	Support
CM1	Churn Detection machines	6.4.17	As defined in Figure 6-23 and Figure 6-24	CM:M	N/A [ ] Yes [ ]

A.2.24 Marker protocol

Item	Feature	Subclause	Value/Comment	Status	Support
FP1	Respond to all received Marker PDUs	Item d) in 5.3, 6.5.1	As specified by 6.5.4	!DRNI1:M DRNI1:O	N/A [ ] Yes [ ] No [ ]
FP2	Use of the Marker protocol	6.5.1	As specified by 6.5.4	O	Yes [ ] No [ ]
FP3	MARKER.request service primitives request rate	6.5.4.1	Maximum of five during any one-second period	MG:M	N/A [ ] Yes [ ]
FP6	Marker PDU structure	6.5.3.3	As shown in Figure 6-27 and as described	MG:M	N/A [ ] Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
FP7	Marker Response PDU structure	Item d) in 5.3, 6.5.3.3	As shown in Figure 6-27 and as described	!DRNI1:M DRNI1:O	N/A [ ] Yes [ ] No [ ]
FP8	Marker Responder state machine	Item d) in 5.3, 6.5.4.2	As specified in Figure 6-28 and 6.5.4.2.1 through 6.5.4.2.2	!DRNI1:M DRNI1:O	N/A [ ] Yes [ ] No [ ]
FP9	Validation of Marker Request PDUs	Item d) in 5.3, 6.5.4.2.2	Marker Responder shall not validate the Version Number, Pad, or Reserved fields	!DRNI1:M DRNI1:O	N/A [ ] Yes [ ] No [ ]

A.2.25 Management

Item	Feature	Subclause	Value/Comment	Status	Support
	If MGT is not supported, mark N/A and ignore the remainder of this table				N/A [ ]
MGT1	Is the Basic package supported?	Table 7–1, 7.3.1.1.1, 7.3.2.1.1	Support of the Managed objects marked as members of the Basic package in Table 7–1	MGT: M	N/A [ ] Yes [ ]
MGT2	Is the Mandatory package supported?	Table 7–1, 7.3.1.1.2, 7.3.1.1.3, 7.3.1.1.4, 7.3.1.1.5, 7.3.1.1.6, 7.3.1.1.7, 7.3.1.1.8, 7.3.1.1.9, 7.3.1.1.10, 7.3.1.1.11, 7.3.1.1.12, 7.3.1.1.13, 7.3.1.1.14, 7.3.1.1.15, 7.3.1.1.16, 7.3.1.1.19, 7.3.1.1.20, 7.3.1.1.31, 7.3.1.2.1, 7.3.1.2.2, 7.3.1.1.32, 7.3.2.1.2, 7.3.2.1.3, 7.3.2.1.4, 7.3.2.1.5, 7.3.2.1.6, 7.3.2.1.7, 7.3.2.1.8, 7.3.2.1.9, 7.3.2.1.10, 7.3.2.1.11, 7.3.2.1.12, 7.3.2.1.13, 7.3.2.1.14, 7.3.2.1.15, 7.3.2.1.16, 7.3.2.1.17, 7.3.2.1.18, 7.3.2.1.19, 7.3.2.1.20, 7.3.2.1.21, 7.3.2.1.22, 7.3.2.1.23, 7.3.2.1.24, 7.3.2.2.1, 7.3.2.2.1	Support of the Managed objects marked as members of the Mandatory package in Table 7–1	MGT: M	N/A [ ] Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
MGT3	Is the Recommended package supported?	Table 7–1, 7.3.1.1.17, 7.3.1.1.18, 7.3.1.1.25, 7.3.1.1.26, 7.3.1.1.27, 7.3.1.1.28, 7.3.1.1.29, 7.3.1.1.30	Support of the Managed objects marked as members of the Recommended package in Table 7–1	MGT: O	N/A [ ] Yes [ ] No [ ]
MGT4	Is the Optional package supported?	Table 7–1, 7.3.1.1.21, 7.3.1.1.22, 7.3.1.1.23, 7.3.1.1.24	Support of the Managed objects marked as members of the Optional package in Table 7–1	MGT: O	N/A [ ] Yes [ ] No [ ]
MGT5	Is the Aggregation Port Statistics package supported?	Table 7–1, 7.3.3.1.1, 7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4, 7.3.3.1.5, 7.3.3.1.6, 7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9	Support of the Managed objects marked as members of the Aggregation Port Statistics package in Table 7–1	MGT: O	N/A [ ] Yes [ ] No [ ]
MGT6	Is the Aggregation Port Debug Information package supported?	Table 7–1, 7.3.4.1.1, 7.3.4.1.2, 7.3.4.1.3, 7.3.4.1.4, 7.3.4.1.5, 7.3.4.1.6, 7.3.4.1.7, 7.3.4.1.8, 7.3.4.1.9, 7.3.4.1.10, 7.3.4.1.11, 7.3.4.1.12, 7.3.4.1.13	Support of the Managed objects marked as members of the Aggregation Port Debug Information package in Table 7–1	MGT: O	N/A [ ] Yes [ ] No [ ]
MGT7	Is the Per-Service Frame Distribution package supported?	Table 7–1, 7.3.1.1.33, 7.3.1.1.34, 7.3.1.1.35, 7.3.1.1.36, 7.3.1.1.37, 7.3.1.1.38, 7.3.1.1.39 7.3.2.1.25, 7.3.2.1.26, 7.3.2.1.27, 7.3.2.1.28, 7.3.2.1.29	Support of the Managed objects marked as members of the Per-Service Frame Distribution package in Table 7–1	MGT AND PSFD: M	N/A [ ] Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
MGT8	Are the CDS Churn Detection managed objects supported?	Table 7–1, 7.3.4.1.14, 7.3.4.1.15, 7.3.4.1.16, 7.3.4.1.17	These managed object are applicable only when Conversation-sensitive frame collection and distribution as specified in 6.6 is supported	MGT AND PSFD: O MGT6 AND PSFD: M	N/A [ ] Yes [ ] No [ ]
MGT9	Is the DRNI package supported?	Table 7–1, 7.4.1.1.1, 7.4.1.1.2, 7.4.1.1.3, 7.4.1.1.4, 7.4.1.1.5, 7.4.1.1.6, 7.4.1.1.7, 7.4.1.1.8, 7.4.1.1.9, 7.4.1.1.10, 7.4.1.1.11, 7.4.1.1.12, 7.4.1.1.13, 7.4.1.1.14, 7.4.1.1.15, 7.4.1.1.16, 7.4.1.1.20, 7.4.1.1.21, 7.4.1.1.22, 7.4.1.1.23, 7.4.2, 7.4.2.1.1, 7.4.2.1.2, 7.4.2.1.3, 7.4.2.1.4, 7.4.2.1.5, 7.4.2.1.6	Support of the Managed objects marked as members of the DRNI package in Table 7–1	MGT AND DRNI: M	N/A [ ] Yes [ ]
MGT11	Is the aDrniEncapsulationMethod managed object supported?	7.4.1.1.17	This managed object is applicable only when Network / IPL sharing by time (9.3.2.1) or Network / IPL sharing by tag (9.3.2.2) or Network / IPL sharing by encapsulation (9.3.2.3) is supported	MGT AND DRNI9:M MGT AND DRNI10:M MGT AND DRNI11:M	N/A [ ] Yes [ ]
MGT12	Is the aDrniIPLEncapMap managed object supported?	7.4.1.1.18	This managed object is applicable only when Network / IPL sharing by tag (9.3.2.2) or Network / IPL sharing by encapsulation (9.3.2.3) is supported	MGT AND DRNI10:M MGT AND DRNI11:M	N/A [ ] Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
MGT13	Is the aDrniNetEncapMap managed object supported?	7.4.1.1.19	This managed object is applicable only when Network / IPL sharing by tag (9.3.2.2) is supported	MGT AND DRNI10:M	N/A [ ] Yes [ ]
MGT14	Is the IPP statistics package supported?	Table 7–1, 7.4.3.1.1, 7.4.3.1.2, 7.4.3.1.3, 7.4.3.1.4	Support of the Managed objects marked as members of the IPP statistics package in Table 7–1	MGT AND DRNI: O	N/A [ ] Yes [ ] No [ ]
MGT15	Is the IPP debug information package supported?	Table 7–1, 7.4.4.1.1, 7.4.4.1.2, 7.4.4.1.3, 7.4.4.1.4	Support of the Managed objects marked as members of the IPP debug information package in Table 7–1	MGT AND DRNI: O	N/A [ ] Yes [ ] No [ ]

A.2.26 Per-Service Frame Distribution

Item	Feature	Subclause	Value/Comment	Status	Support
	If PSFD is not supported, mark N/A and ignore the remainder of this table				N/A [ ]
PSFD1	Is Frame Distribution by VID supported?	8.2.2		PSFD: M	N/A [ ] Yes [ ]
PSFD2	Is Frame Distribution by I-SID supported?	8.2.2, 8.2.3	Relevant only if Per-service frame distribution is implemented	PSFD: O	N/A [ ] Yes [ ] No [ ]
PSFD3	Does the implementation support Frame Distribution by other methods not specified by this standard?			PSFD: O	N/A [ ] Yes [ ] No [ ]

A.2.27 Conversation-sensitive frame collection and distribution

Item	Feature	Subclause	Value/Comment	Status	Support
	If CSCD is not supported, mark N/A and ignore the remainder of this table				N/A [ ]
CSCD1	Are the Conversation-sensitive collection and distribution state diagrams supported?	6.6.1	As shown in Figure 6-29 and as described	CSCD: M	Yes [ ]
CSCD2	Does the function DeterminePortConversationID involve the application of local Service ID to Conversation ID mappings?	6.6.1.1.3, 7.3.1.1.8	Mapping ability required for classification by I-SID	CSCD:O CSCD AND PSFD2: M	Yes [ ] No [ ]
CSCD3	Are the Verification state diagram (VER), the Report for Management Action state diagram (RMA), the Receive Long LACPDU state diagram (RXL) and the Update Mask state diagram (UM), the Actor CDS Churn Detection machine state diagram and the Partner CDS Churn Detection machine state diagram supported?	6.6.2	As shown in Figure 6-31, Figure 6-32, Figure 6-33, Figure 6-34, Figure 6-35, Figure 6-36, and as described	CSCD: O	Yes [ ] No [ ]

A.2.28 Configuration capabilities and restrictions

Item	Feature	Subclause	Value/Comment	Status	Support
CCR1	Algorithm used to determine subset of Aggregation Ports that will be aggregated in Systems that have limited aggregation capability	6.7.1	As specified in items a) to e) of 6.7.1	M	Yes[ ]
CCR2	Key value modification to generate optimum aggregation	6.7.2		O	Yes [ ] No [ ]
CCR3	Key value modification when System has higher System Aggregation Priority			CCR2:M	N/A [ ] Yes [ ]
CCR4	Key value modification when System has lower System Aggregation Priority			CCR2:X	N/A [ ] No [ ]

A.2.29 Link Aggregation on shared-medium links

Item	Feature	Subclause	Value/Comment	Status	Support
LSM1	Shared-medium links—Configuration	6.7.3	Configured as Individual links	M	Yes [ ]
LSM2	Shared-medium links—Operation of LACP	6.7.3	LACP is disabled	M	Yes [ ]

A.2.30 Distributed Resilient Network Interconnect

Item	Feature	Subclause	Value/Comment	Status	Support
	If DRNI is not supported, mark N/A and ignore the remainder of this table				N/A [ ]
DRNI1	Is the DR Function and the emulation of a Distributed Relay in cooperation with a single other Portal System, which also conforms to the provisions of this standard for DRNI, supported?	Item b1) in 5.4, 9.2, 9.3	Support the state machines and procedures in 9.3 for a DR Function, constrained by the presence of up to a single IPL and up to two Portal Systems in the Portal, as specified	DRNI:M	Yes [ ]
DRNI2	Is the DR Function and the emulation of a Distributed Relay, in cooperation with two other Portal Systems, which also conform to the provisions of this standard for DRNI, supported?	Item a1) in 5.4.1, 9.2, 9.3	Support the state machines and procedures in 9.3 for a DR Function, for a Portal of 3 Portal Systems, as specified	DRNI:O	Yes [ ] No [ ]
DRNI3	Is the DRCP supported for Portal of two Portal Systems?	Item b2) in 5.4, 9.4	Support the state machines and procedures in 9.4 as constrained by the presence of up to a single IPL and up to two Portal Systems in the Portal	DRNI:M	Yes [ ]
DRNI4	Is the DRCP supported for a Portal of three Portal Systems?	Item a2) in 5.4.1, 9.4	Support the state machines and procedures in 9.4 for a Portal of 3 Portal Systems	DRNI2:M	Yes [ ] N/A [ ]
DRNI5	Is a Portal Topology of three Portal Systems in a ring connected by three IPLs supported?	9.3.1, 9.4		DRNI4:O	Yes [ ] No [ ] N/A [ ]
DRNI6	Are DRCPDUs supported?	Item b3) in 5.4, 9.4.2	Transmit and receive DRCPDUs in the formats specified in 9.4.2 and 9.4.3	DRNI:M	Yes [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
DRNI7	Can the IPL supported by using separate physical links for IPLs and for the network links?	Item c) in 5.4, 9.3.2	Support separate physical links for IPLs and network links, as specified in item a)	DRNI:M	Yes [ ]
DRNI8	Can the IPL be supported by a LAG and thus consisting of a number of physical links?	Item b) in 5.4.1, 9.3.2	Support a separate Aggregation Port for the IPL, as specified in item b)	DRNI:O	Yes [ ] No [ ]
DRNI9	Can the IPL be supported by the Network / IPL sharing by time method?	Item b) in 5.4.1, 9.3.2.1		DRNI:O	Yes [ ] No [ ]
DRNI10	Can the IPL be supported by the Network / IPL sharing by tag method?	Item b) in 5.4.1, 9.3.2.2		DRNI:O	Yes [ ] No [ ]
DRNI11	Can the IPL be supported by the Network / IPL sharing by encapsulation method?	Item b) in 5.4.1, 9.3.2.3		DRNI:O	Yes [ ] No [ ]

A.2.31 DRCPDU structure

Item	Feature	Subclause	Value/Comment	Status	Support
	If DRNI6 is not supported, mark N/A and ignore the remainder of this table				N/A [ ]
DRST1	DRCPDU addressing and protocol identifications	9.4.3	As described	DRNI6:M	Yes [ ]
DRST2	DRCPDU structure	9.4.2	As shown in Figure 9-9 and as described	DRNI6:M	Yes [ ]
DRST3	DRCPDU structure	9.4.3	All Reserved octets ignored on receipt and transmitted as zero	DRNI6:M	Yes [ ]
DRST4	Are the Terminator TLV, the Portal Information TLV, the Portal Configuration the Information TLV, the DRCP State TLV, the Home Ports Information TLV, the Neighbor Ports Information TLV, the Home Gateway Vector TLV, the Neighbor Gateway Vector TLV and the Conversation Vector TLVs supported?	9.4.3, 9.4.3.2, 9.4.3.3	As shown in Figure 9-9, Figure 9-10, Figure 9-11, Figure 9-12, Figure 9-13, Figure 9-14, Figure 9-15, Figure 9-16, Figure 9-17, Figure 9-18, and as described	DRNI6:M	Yes [ ]
DRST5	Is the ONN bit in the Topology field within the Portal Configuration Information TLV supported?	Item a3) in 5.4.1, 9.4.3	As shown in Figure 9-9 and as described	DRNI1:O DRNI2:M	Yes [ ] No [ ] N/A [ ]

Item	Feature	Subclause	Value/Comment	Status	Support
DRST6	Is the Other Gateway bit in the DRCP State TLV supported?	Item a3) in 5.4.1, 9.4.3	As shown in Figure 9-9 and as described	DRNI1:O DRNI2:M	Yes [ ] No [ ] N/A [ ]
DRST7	Is the Other Ports Information TLV and the Other Gateway Vector TLV supported?	Item a3) in 5.4.1, 9.4.3	As shown in Figure 9-9 and as described	DRNI1:O DRNI2:M	Yes [ ] No [ ] N/A [ ]
DRST8	Is the Network/IPL Sharing Method TLV supported?	Item b) in 5.4.1, 9.4.3.4.1	As shown in Figure 9-19 and as described	DRNI9:M DRNI10:M DRNI11:M	Yes [ ] N/A [ ]
DRST9	Is the Network/IPL Sharing Encapsulation TLV supported?	Item b) in 5.4.1, 9.4.3.4.2	As shown in Figure 9-20 and as described	DRNI10:M DRNI11:M	Yes [ ] N/A [ ]
DRST10	Is the Organization-Specific TLV supported?	Item c) in 5.4.1, 9.4.3.5	As shown in Figure 9-9 and as described	DRNI1:O DRNI2:M	Yes [ ] No [ ] N/A [ ]

A.2.32 Bridge specific support

Item	Feature	Subclause	Value/Comment	Status	Support
BRG	Is the System implementing Link Aggregation is an IEEE 802.1Q Bridge? If not mark N/A and ignore the remainder of this table			O	Yes [ ] N/A [ ]
BRG1	Is Per I-SID frame distribution supported on PIPs (6.10 of IEEE Std 802.1Q-2014) or CBPs (6.11 of IEEE Std 802.1Q-2014)	8.2.2, 8.2.3		BRG AND PSFD:M	Yes [ ] N/A [ ]
BRG2	Is MAC Address Synchronization supported?	Item b) in 5.4.1, Annex G	Only required when the IPL is supported by the Network / IPL sharing by time method and the Per-service frame distribution is not used	BRG AND DRNI9:M	Yes [ ] N/A [ ]

Annex B

(informative)

Collection and distribution algorithms

B.1 Introduction

The specification of the Frame Collection and Frame Distribution functions was defined with the following considerations in mind:

a) Frame duplication is not permitted.
b) Frame ordering has to be preserved in aggregated links. Strictly, the Internal Sublayer Service specification (IEEE Std 802.1AC) states that order has to 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 Frame Collection function that is independent of the distribution algorithm(s) employed by the Partner System.
d) In the interests of simplicity and scalability, the Frame Collection function should not perform reassembly functions, reorder received frames, or modify received frames. Frame Distribution functions, therefore, do not make use of segmentation techniques, do not label or otherwise modify transmitted frames in any way, and have to operate in a manner that will inherently ensure proper ordering of received frames with the specified Frame Collector.
e) The distribution and collection algorithms 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 algorithm will be applicable in all cases.

A simple Frame Collection function has been specified. The Frame Collector preserves the order of frames received on a given link, but does not preserve frame ordering among links. The Frame 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 Aggregator Client at a later time.

Given the wide variety of potential distribution algorithms, the normative text in Clause 6 specifies only the requirements that such algorithms have to 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 (6.5) 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 6 are met.

B.2 Port selection

A distribution algorithm selects the Aggregation Port used to transmit a given frame, such that the same Aggregation 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 Aggregation Port in the case of a bridge.

The algorithm may assign one or more conversations to the same Aggregation Port; however, it has to not allocate some of the frames of a given conversation to one Aggregation Port and the remainder to different Aggregation Ports. The information used to assign conversations to Aggregation Ports could include the following:

a) Source MAC address
b) Destination MAC address
c) Reception Aggregation Port
d) 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) Aggregation Port selection across a given number of Aggregation 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 Aggregation Ports for all traffic models, it might be appropriate to weight the Aggregation Port selection in favor of Aggregation Ports that are carrying lower traffic levels. In more sophisticated approaches, load balancing is dynamic; i.e., the Aggregation Port selected for a given set of conversations changes over time, independent of any changes that take place in the membership of the aggregation.

B.3 Dynamic reallocation of conversations to different Aggregation Ports

It may be necessary for a given conversation or set of conversations to be moved from one Aggregation Port to one or more others, as a result of

a) An existing Aggregation Port being removed from the aggregation,
b) A new Aggregation Port being added to the aggregation, or
c) A decision on the part of the Frame Distributor to redistribute the traffic across the set of Aggregation Ports.

Before moving conversation(s) to a new Aggregation 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 (6.5) can be used to ensure that no misordering of frames occurs:

1) Stop transmitting frames for the set of conversations affected. If the Aggregator 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 Aggregation 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 Aggregation 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 6.2.3) defines the maximum delay that the Frame Collector can introduce between receiving a frame from an Aggregation Port and either delivering it to the Aggregator 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 Frame Collector state machine is concerned, CollectorMaxDelay is a constant; however, a management attribute, aAggCollectorMaxDelay (7.3.1.1.32), 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 Frame Collector is less than the resolution of the parameter (10 µs). In this case, the delay that has to 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 preconfigure 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 Frame Distributor and the remote Frame Collector. The timeout described recovers from loss of Marker or Marker Response PDUs that can occur.

B.4 Topology considerations in the choice of distribution algorithm

Figure B.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 Bridge 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 Bridge 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 algorithm is independent of the distribution algorithm(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 Bridges 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 Bridges could revert to MAC Address-based allocation.

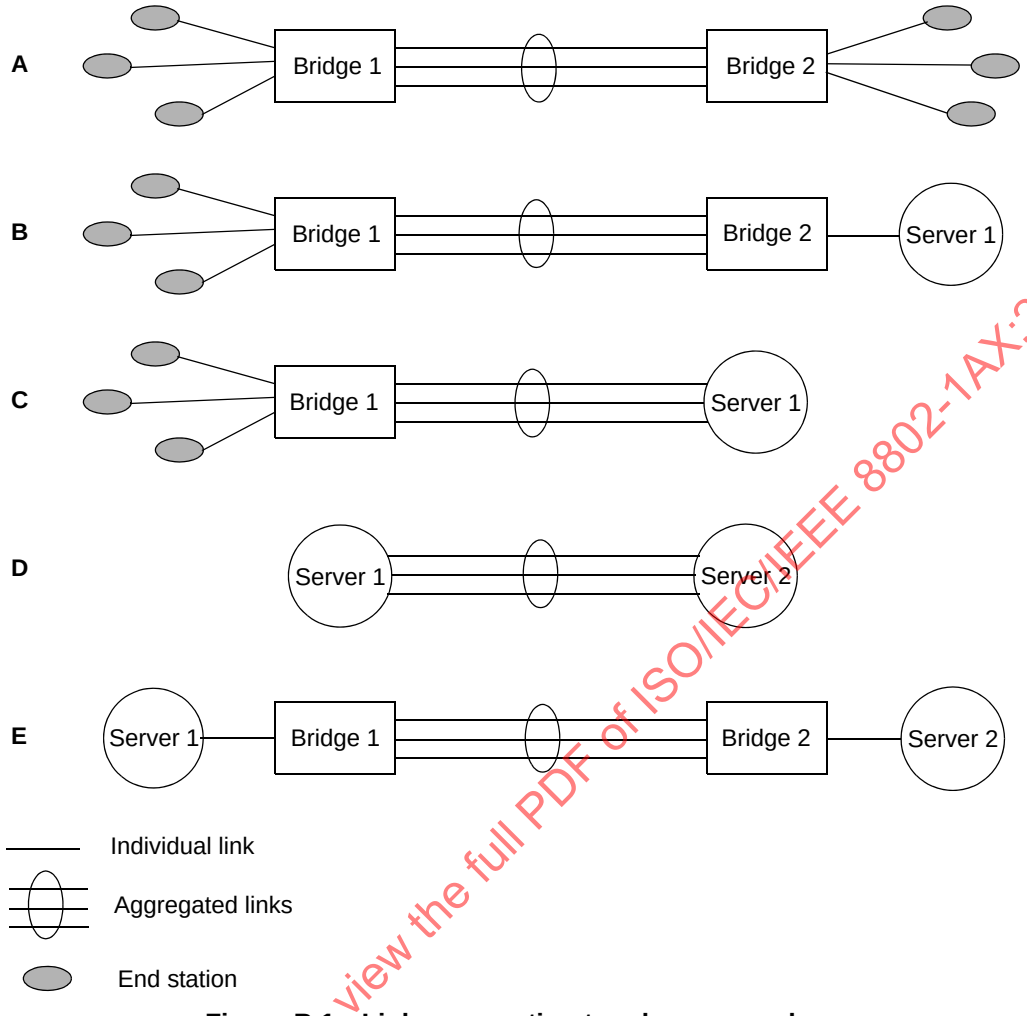


Figure B.1—Link aggregation topology examples

Annex C

(informative)

LACP standby link selection and dynamic Key management

C.1 Introduction

While any two Aggregation 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 Aggregation 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 Aggregation Ports attached to a single Aggregator, or the particular way more than two Aggregation 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 (LAG). For this to happen, one or another of the communicating Systems has to change the operational Key values used for the Aggregation 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.

C.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 Aggregation Ports that might be imposed by other Systems.
d) Correct configuration of parameters, i.e., they retain the plug and play attributes of LACP.

C.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.

Aggregation Ports are considered for active use in an aggregation in link priority order, starting with the Aggregation Port attached to the highest priority link. Each Aggregation Port is selected for active use if preceding higher priority selections can also be maintained; otherwise, the Aggregation Port is selected as standby.

C.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 LAG 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.

C.5 A dynamic Key management algorithm

The following algorithm is simple but effective.

After the “settling time” (see C.4) has elapsed, the controlling System scans its Aggregation Ports in the LAG (i.e., all those Aggregation Ports with a specific operational Key value that have the same Partner System Priority, System ID, and Key) in descending priority order.

For each Aggregation Port, it may wish to know

a) Is the Aggregation Port (i.e., the Actor) *capable* of being aggregated with the Aggregation Ports already selected for aggregation with the current Key? Alternatively, is the Actor *not capable* of this aggregation?
b) Is the Aggregation Port's Partner IN_SYNC or is the Partner OUT_OF_SYNC?

As it inspects each Aggregation Port, it may

c) *Select* the Aggregation 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 Aggregation 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 Aggregation Ports in the current LAG that have their Keys changed will be changed to this new value.

As the Aggregation Ports are scanned for the first time

1) The highest priority Aggregation Port is always selected.
   If it is capable and IN_SYNC, move to step 2).
   Otherwise, **change** the operational Key of all other Aggregation Ports (if any) in this LAG, and apply this dynamic Key algorithm to those Aggregation Ports, beginning with step 1), after the settling time.
2) Move to the next Aggregation Port.
   If there is a next Aggregation Port, continue at step 3).
   Otherwise, dynamic Key changes for Aggregation Ports with this operational Key are complete.
   Note that Aggregation 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 Aggregation Ports, beginning with step 1), after the settling time.
3) If this Aggregation Port is capable and IN_SYNC:
   select it, and repeat from step 2).
   If this Aggregation Port is OUT_OF_SYNC:
   change the operational Key, and repeat from step 2).
   If this Aggregation Port is not capable but IN_SYNC:
   change the operational Key, move to step 4).
4) Move to the next Aggregation Port.
   If there is a next Aggregation Port, continue at step 5).
   Otherwise if there are still Aggregation Ports in the current LAG (which will have the current operational Key), wait for the settling time and apply the dynamic Key management algorithm, beginning with the first such Aggregation Port, at step 3).
   Otherwise, dynamic Key changes for Aggregation Ports with this operational Key are complete.
5) **If** this Aggregation 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.

C.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 C.1. System A is the higher priority System.

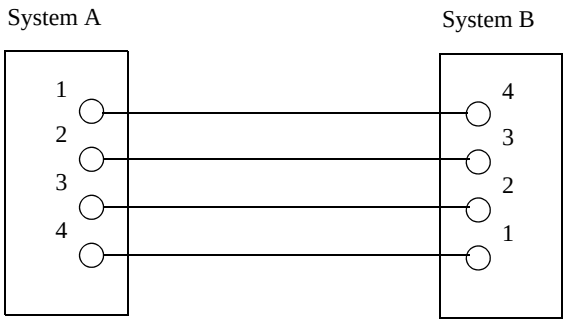


Figure C.1—Example 1

The administrative Key for all of System A and System B's Aggregation Ports is 1. Neither System knows before the configuration is chosen that all its Aggregation 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 Aggregation 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 Aggregation 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 s (depending on how fast System B completes the process of attaching its Aggregation Ports to the selected Aggregator) after the connections were first made, and both aggregations could be operating within 5 s.

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.

C.7 Example 2

A System has the odd design constraint that each of its four Aggregation Ports may be aggregated with one other as follows:

a) Aggregation Port 1 with Aggregation Port 2, or Aggregation Port 4.
b) Aggregation Port 2 with Aggregation Port 3, or Aggregation Port 1.
c) Aggregation Port 3 with Aggregation Port 4, or Aggregation Port 2.
d) Aggregation Port 4 with Aggregation Port 1, or Aggregation Port 3.

This is equivalent to each Aggregation Port being able to aggregate with either neighbor, understanding the Aggregation Ports to be arranged in a circle.

Two such Systems are connected with four parallel links as shown in Figure C.2.

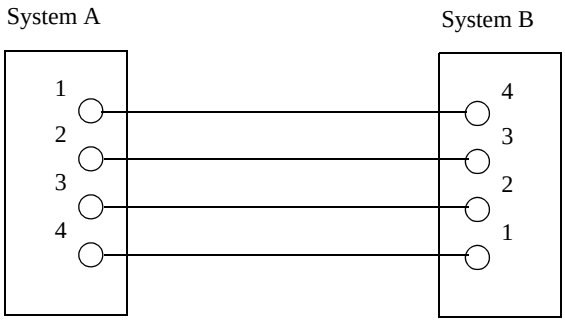


Figure C.2—Example 2a

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 C.3, the following would occur, assuming that System A operates the simple algorithm already described:

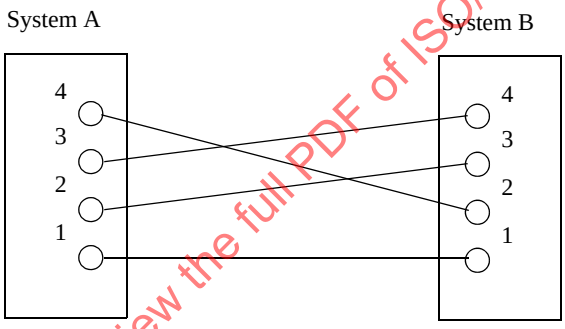


Figure C.3—Example 2b

Initially System A advertises an operational Key equal to the administrative Key value of 1 on all Aggregation 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 Aggregation Port standby. System B can aggregate B4-A3, so the Aggregation 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 Aggregation Ports 2 and 3, retaining the initial Key of 1 for Aggregation 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 D

(normative)

SMIv2 MIB definitions for Link Aggregation[10]

D.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 this standard. This annex includes an MIB module that is SNMPv2 SMI compliant.

D.2 SNMP Management Framework

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

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB modules are defined using the mechanisms defined in the Structure of Management Information (SMI). This clause specifies a MIB module that is compliant to the SMIv2, which is described in IETF STD 58, comprising IETF RFC 2578 (1999), IETF RFC 2579 (1999), and IETF RFC 2580 (1999).

D.3 Security considerations

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

This MIB module relates to systems that provide resilient transport at very high capacities based on link aggregation. Improper manipulation of the following tables and the objects in the tables may cause network instability and result in loss of service for a large number of end users.

dot3adAggTable
dot3adAggXTable
dot3adAggPortTable
dot3adAggPortXTable
dot3adAggPortSecondXTable

In particular, the object dot3adAggPortProtocolDA of the dot3adAggPortXTable sets the DA for LACP frames and could be manipulated by an attacker to force a misconfiguration error causing the LAG to fail.

[10]*Copyright release for SMIv2 MIB:* Users of this standard may freely reproduce the SMIv2 MIB in this annex so it can be used for its intended purpose.

Manipulation of the following objects can cause frames to be directed to the wrong ports, which can cause loss of connectivity due to excessive congestion losses, as well as spoiling some or all of the goals of Clause 8, such as bidirectional congruity.

dot3adAggConversationAdminPortTable
dot3adAggAdminServiceConversationMapTable

For a system implementing Distributed Resilient Network Interconnects (DRNIs), improper manipulation of the following tables and the objects in the tables can have similar negative effects.

dot3adDrniTable
dot3adDrniConvAdminGatewayTable
dot3adDrniIPLEncapMapTable
dot3adDrniNetEncapMapTable
dot3adIPPAttributeTable

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables whose objects contain information that may be used to gain sensitive information, which may be used to damage the organization providing the link aggregation services or the users of those services. Sensitive information could be used by an attacker to determine which attacks might be useful to attempt against a given device or to detect whether attacks are being blocked or filtered.

dot3adAggPortStatsTable
dot3adAggPortDebugTable
dot3adAggPortDebugXTable
dot3adIPPDebugTable

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

Implementations SHOULD provide the security features described by the SNMPv3 framework (see IETF RFC 3410), and implementations claiming compliance to the SNMPv3 standard MUST include full support for authentication and privacy via the User-based Security Model (USM) IETF RFC 3414 with the AES cipher algorithm IETF RFC 3826. Implementations MAY also provide support for the Transport Security Model (TSM) IETF RFC 5591 and the View-based Access Control Model IETF RFC 3415 in combination with a secure transport such as SSH IETF RFC 5592 or TLS/DTLS IETF RFC 6353.

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

D.4 Structure of the MIB module

A single MIB module is defined in this annex. Objects in the MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. The overall structure and assignment of objects to their subtrees and their corresponding definitions in Clause 7 is shown in the following subclauses.

D.4.1 Relationship to the managed objects defined in Clause 7

This subclause contains cross references to the objects defined in Clause 7. Table D.1 contains cross references for the MIB module objects defined in this annex. Table D.2 contains definitions of ifTable elements, as defined in IETF RFC 2863, for an Aggregator. These table elements are cross referenced to the corresponding definitions in Clause 7.

Table D.1—Managed object cross reference table

Definition in Clause 7	MIB object
7.3.1.1.1 aAggID	ifIndex value
7.3.1.1.4 aAggActorSystemID	dot3adAggActorSystemID
7.3.1.1.5 aAggActorSystemPriority	dot3adAggActorSystemPriority
7.3.1.1.6 aAggAggregateOrIndividual	dot3adAggAggregateOrIndividual
7.3.1.1.7 aAggActorAdminKey	dot3adAggActorAdminKey
7.3.1.1.8 aAggActorOperKey	dot3adAggActorOperKey
7.3.1.1.10 aAggPartnerSystemID	dot3adAggPartnerSystemID
7.3.1.1.11 aAggPartnerSystemPriority	dot3adAggPartnerSystemPriority
7.3.1.1.12 aAggPartnerOperKey	dot3adAggPartnerOperKey
7.3.1.1.30 aAggPortList	dot3adAggPortListTable
7.3.1.1.32 aAggCollectorMaxDelay	dot3adAggCollectorMaxDelay
7.3.1.1.33 aAggPortAlgorithm	dot3adAggPortAlgorithm
7.3.1.1.34 aAggPartnerAdminPortAlgorithm	dot3adAggPartnerAdminPortAlgorithm
7.3.1.1.35 aAggConversationAdminLink[]	dot3adAggConversationAdminLinkTable
7.3.1.1.36 aAggPartnerAdminPortConversationListDigest	dot3adAggPartnerAdminPortConversationListDigest
7.3.1.1.37 aAggAdminDiscardWrongConversation	dot3adAggAdminDiscardWrongConversation
7.3.1.1.38 aAggAdminServiceConversationMap[]	dot3adAggAdminServiceConversationMapTable
7.3.1.1.39 aAggPartnerAdminConvServiceMappingDigest	dot3adAggPartnerAdminConvServiceMappingDigest
7.3.2.1.1 aAggPortID	ifIndex value
7.3.2.1.2 aAggPortActorSystemPriority	dot3adAggPortActorSystemPriority
7.3.2.1.3 aAggPortActorSystemID	dot3adAggPortActorSystemID
7.3.2.1.4 aAggPortActorAdminKey	dot3adAggPortActorAdminKey
7.3.2.1.5 aAggPortActorOperKey	dot3adAggPortActorOperKey
7.3.2.1.6 aAggPortPartnerAdminSystemPriority	dot3adAggPortPartnerAdminSystemPriority
7.3.2.1.7 aAggPortPartnerOperSystemPriority	dot3adAggPortPartnerOperSystemPriority
7.3.2.1.8 aAggPortPartnerAdminSystemID	dot3adAggPortPartnerAdminSystemID
7.3.2.1.9 aAggPortPartnerOperSystemID	dot3adAggPortPartnerOperSystemID
7.3.2.1.10 aAggPortPartnerAdminKey	dot3adAggPortPartnerAdminKey

Table D.1—Managed object cross reference table *(continued)*

Definition in Clause 7	MIB object
7.3.2.1.11 aAggPortPartnerOperKey	dot3adAggPortPartnerOperKey
7.3.2.1.12 aAggPortSelectedAggID	dot3adAggPortSelectedAggID
7.3.2.1.13 aAggPortAttachedAggID	dot3adAggPortAttachedAggID
7.3.2.1.14 aAggPortActorPort	dot3adAggPortActorPort
7.3.2.1.15 aAggPortActorPortPriority	dot3adAggPortActorPortPriority
7.3.2.1.16 aAggPortPartnerAdminPort	dot3adAggPortPartnerAdminPort
7.3.2.1.17 aAggPortPartnerOperPort	dot3adAggPortPartnerOperPort
7.3.2.1.18 aAggPortPartnerAdminPortPriority	dot3adAggPortPartnerAdminPortPriority
7.3.2.1.19 aAggPortPartnerOperPortPriority	dot3adAggPortPartnerOperPortPriority
7.3.2.1.20 aAggPortActorAdminState	dot3adAggPortActorAdminState
7.3.2.1.21 aAggPortActorOperState	dot3adAggPortActorOperState
7.3.2.1.22 aAggPortPartnerAdminState	dot3adAggPortPartnerAdminState
7.3.2.1.23 aAggPortPartnerOperState	dot3adAggPortPartnerOperState
7.3.2.1.24 aAggPortAggregateOrIndividual	dot3adAggPortAggregateOrIndividual
7.3.2.1.25 aAggPortOperConversationPasses	dot3adAggPortOperConversationPasses
7.3.2.1.26 aAggPortOperConversationCollected	dot3adAggPortOperConversationCollected
7.3.2.1.27 aAggPortLinkNumberID	dot3adAggPortLinkNumberId
7.3.2.1.28 aAggPortPartnerAdminLinkNumberID	dot3adAggPortPartnerAdminLinkNumberId
7.3.2.1.29 aAggPortWTRTime	dot3adAggPortWTRTime
7.3.2.2.1 aAggPortProtocolDA	dot3adAggPortProtocolDA
7.3.3.1.1 aAggPortStatsID	ifIndex value
7.3.3.1.2 aAggPortStatsLACPDUsRx	dot3adAggPortStatsLACPDUsRx
7.3.3.1.3 aAggPortStatsMarkerPDUsRx	dot3adAggPortStatsMarkerPDUsRx
7.3.3.1.4 aAggPortStatsMarkerResponsePDUsRx	dot3adAggPortStatsMarkerResponsePDUsRx
7.3.3.1.5 aAggPortStatsUnknownRx	dot3adAggPortStatsUnknownRx
7.3.3.1.6 aAggPortStatsIllegalRx	dot3adAggPortStatsIllegalRx
7.3.3.1.7 aAggPortStatsLACPDUsTx	dot3adAggPortStatsLACPDUsTx
7.3.3.1.8 aAggPortStatsMarkerPDUsTx	dot3adAggPortStatsMarkerPDUsTx
7.3.3.1.9 aAggPortStatsMarkerResponsePDUsTx	dot3adAggPortStatsMarkerResponsePDUsTx
7.3.4.1.1 aAggPortDebugInformationID	ifIndex value (see IETF RFC 2863) of the port
7.3.4.1.2 aAggPortDebugRxState	dot3adAggPortDebugRxState
7.3.4.1.3 aAggPortDebugLastRxTime	dot3adAggPortDebugLastRxTime
7.3.4.1.4 aAggPortDebugMuxState	dot3adAggPortDebugMuxState

Table D.1—Managed object cross reference table *(continued)*

Definition in Clause 7	MIB object
7.3.4.1.5 aAggPortDebugMuxReason	dot3adAggPortDebugMuxReason
7.3.4.1.6 aAggPortDebugActorChurnState	dot3adAggPortDebugActorChurnState
7.3.4.1.7 aAggPortDebugPartnerChurnState	dot3adAggPortDebugPartnerChurnState
7.3.4.1.8 aAggPortDebugActorChurnCount	dot3adAggPortDebugActorChurnCount
7.3.4.1.9 aAggPortDebugPartnerChurnCount	dot3adAggPortDebugPartnerChurnCount
7.3.4.1.10 aAggPortDebugActorSyncTransitionCount	dot3adAggPortDebugActorSyncTransitionCount
7.3.4.1.11 aAggPortDebugPartnerSyncTransitionCount	dot3adAggPortDebugPartnerSyncTransitionCount
7.3.4.1.12 aAggPortDebugActorChangeCount	dot3adAggPortDebugActorChangeCount
7.3.4.1.13 aAggPortDebugPartnerChangeCount	dot3adAggPortDebugPartnerChangeCount
7.3.4.1.14 aAggPortDebugActorCDSChurnState	dot3adAggPortDebugActorCDSChurnState
7.3.4.1.15 aAggPortDebugPartnerCDSChurnState	dot3adAggPortDebugPartnerCDSChurnState
7.3.4.1.16 aAggPortDebugActorCDSChurnCount	dot3adAggPortDebugActorCDSChurnCount
7.3.4.1.17 aAggPortDebugPartnerCDSChurnCount	dot3adAggPortDebugPartnerCDSChurnCount
7.4.1.1.1 aDrniID	ifIndex value
7.4.1.1.2 aDrniDescription	dot3adDrniDescription
7.4.1.1.3 aDrniName	dot3adDrniName
7.4.1.1.4 aDrniPortalAddr	dot3adDrniPortalAddr
7.4.1.1.5 aDrniPortalPriority	dot3adDrniPortalPriority
7.4.1.1.6 aDrniThreePortalSystem	dot3adDrniThreePortalSystem
7.4.1.1.7 aDrniPortalSystemNumber	dot3adDrniPortalSystemNumber
7.4.1.1.8 aDrniIntraPortalLinkList	dot3adDrniIntraPortalLinkList
7.4.1.1.9 aDrniAggregator	dot3adDrniAggregator
7.4.1.1.10 aDrniConvAdminGateway[]	dot3adDrniConvAdminGatewayTable
7.4.1.1.11 aDrniNeighborAdminConvGatewayListDigest	dot3adDrniNeighborAdminConvGatewayListDigest
7.4.1.1.12 aDrniNeighborAdminConvPortListDigest	dot3adDrniNeighborAdminConvPortListDigest
7.4.1.1.13 aDrniGatewayAlgorithm	dot3adDrniGatewayAlgorithm
7.4.1.1.14 aDrniNeighborAdminGatewayAlgorithm	dot3adDrniNeighborAdminGatewayAlgorithm
7.4.1.1.15 aDrniNeighborAdminPortAlgorithm	dot3adDrniNeighborAdminPortAlgorithm
7.4.1.1.16 aDrniNeighborAdminDRCPState	dot3adDrniNeighborAdminDRCPState
7.4.1.1.17 aDrniEncapsulationMethod	dot3adDrniEncapsulationMethod
7.4.1.1.18 aDrniIPLEncapMap	dot3adDrniIPLEncapMapTable
7.4.1.1.19 aDrniNetEncapMap	dot3adDrniNetEncapMapTable
7.4.1.1.20 aDrniDRPortConversationPasses	dot3adDrniDRPortConversationPasses

Table D.1—Managed object cross reference table *(continued)*

Definition in Clause 7	MIB object
7.4.1.1.21 aDrniDRGatewayConversationPasses	dot3adDrniDRGatewayConversationPasses
7.4.1.1.22 aDrniPSI	dot3adDrniPSI
7.4.1.1.23 aDrniPortConversationControl	dot3adDrniPortConversationControl
7.4.1.1.24 aDrniIntraPortalPortProtocolDA	dot3adDrniIntraPortalPortProtocolDA
7.4.2.1.2 aIPPPortConversationPasses	dot3adIPPPortConversationPasses
7.4.2.1.3 aIPPGatewayConversationDirection	dot3adIPPGatewayConversationDirection
7.4.2.1.4 aIPPAdminState	dot3adDrniIPPAdminState
7.4.2.1.5 aIPPOperState	dot3adDrniIPPOperState
7.4.2.1.6 aIPPTimeOfLastOperChange	dot3adDrniIPPTimeOfLastOperChange
7.4.3.1.1 aIPPStatsID	ifIndex value
7.4.3.1.2 aIPPStatsDRCPDUsRx	dot3adDrniIPPStatsDRCPDUsRx
7.4.3.1.3 aIPPStatsIllegalRx	dot3adDrniIPPStatsIllegalRx
7.4.3.1.4 aIPPStatsDRCPDUsTx	dot3adDrniIPPStatsDRCPDUsTx
7.4.4.1.1 aIPPDebugInformationID	ifIndex value
7.4.4.1.2 aIPPDebugDRCPRxState	dot3adDrniIPPDebugDRCPRxState
7.4.4.1.3 aIPPDebugLastRxTime	dot3adDrniIPPDebugLastRxTime
7.4.4.1.4 aIPPDebugDifferPortalReason	dot3adIPPDebugDifferPortalReason

Table D.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 2863.
ifDescr	Interpreted as defined in IETF RFC 2863 and as further refined in the definition of aAggDescription (7.3.1.1.2).
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 (7.3.1.1.16).
ifPhysAddress	The individual MAC Address of the Aggregator as defined for aAggMACAddress (7.3.1.1.9).
ifAdminStatus	The administrative state of the Aggregator as defined for aAggAdminState (7.3.1.1.13).
ifOperStatus	The operational state of the Aggregator as defined for aAggOperState (7.3.1.1.14).

Table D.2—ifTable element definitions for an Aggregator *(continued)*

Object	Definition
ifLastChange	Interpreted as defined in IETF RFC 2863; see also the definition of aAggTimeOfLastOperChange (7.3.1.1.15).
ifInOctets	The total number of user data octets received by the aggregation, as defined for aAggOctetsRxOK (7.3.1.1.18).
ifInUcastPkts	The total number of unicast user data frames received by the aggregation. This value is calculated as the value of aAggFramesRxOK (7.3.1.1.20), less the values of aAggMulticastFramesRxOK (7.3.1.1.22) and aAggBroadcastFramesRxOK (7.3.1.1.24).
ifInNUcastPkts	Deprecated in IETF RFC 2863.
ifInDiscards	The number of frames discarded on reception, as defined for aAggFramesDiscardedOnRx (7.3.1.1.26).
ifInErrors	The number of frames with reception errors, as defined for aAggFramesWithRxErrors (7.3.1.1.28).
ifInUnknownProtos	The number of unknown protocol frames discarded on reception, as defined for aAggUnknownProtocolFrames (7.3.1.1.29).
ifOutOctets	The total number of user data octets transmitted by the aggregation, as defined for aAggOctetsTxOK (7.3.1.1.17).
ifOutUcastPkts	The total number of unicast user data frames transmitted by the aggregation. This value is calculated as the value of aAggFramesTxOK (7.3.1.1.19), less the values of aAggMulticastFramesTxOK (7.3.1.1.21) and aAggBroadcastFramesTxOK (7.3.1.1.23).
ifOutNUcastPkts	Deprecated in IETF RFC 2863.
ifOutDiscards	The number of frames discarded on transmission, as defined for aAggFramesDiscardedOnTx (7.3.1.1.25).
ifOutErrors	The number of frames discarded due to transmission errors, as defined for aAggFramesWithTxErrors (7.3.1.1.27).
ifOutQLen	Deprecated in IETF RFC 2863. Set to zero if present.
ifSpecific	Deprecated in IETF RFC 2863. Set to { 0.0 } if present.
ifLinkUpDownTrapEnable	See the definition of aAggLinkUpDownNotificationEnable (7.3.1.1.31).
ifConnectorPresent	“FALSE.”
ifHighSpeed	Set to zero.
ifName	The locally assigned textual name of the Aggregator, as defined for aAggName (7.3.1.1.3). Interpreted as defined in IETF RFC 2863.
linkUp TRAP	See the definition of nAggLinkUpNotification (7.3.1.2.1).
linkDown TRAP	See the definition of nAggLinkDownNotification (7.3.1.2.2).

[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.

D.4.2 MIB Subtrees

The correspondence between the objects in the MIB module subtrees and the objects of the managed object classes defined in Clause 7 is described in the following subclauses.

D.4.2.1 The dot3adAgg Subtree

The objects of the dot3adAgg subtree correspond to the objects of the Aggregator managed object class (7.3.1) with the exception of the Aggregator Notifications (7.3.1.2).

D.4.2.2 The dot3adAggPort Subtree

The objects of the dot3adAggPort subtree correspond to the objects of the Aggregation Port managed object class (7.3.2), the Aggregation Port Statistics managed object class (7.3.3) and the Aggregation Port Debug Information managed object class (7.3.4).

D.4.2.3 The dot3adAggNotifications Subtree

The objects of the dot3adAggPort subtree correspond to the Aggregator Notifications (7.3.1.2).

D.4.2.4 The dot3adDrni Subtree

The objects of the dot3adDrni subtree correspond to the objects of the Distributed Relay Managed Object Class (7.4.1).

D.4.2.5 The dot3adIPP Subtree

The objects of the dot3adIPP subtree correspond to the objects of the IPP Managed Objects Class (7.4.2), the IPP Statistics managed object class (7.4.3) and the IPP Debug Information managed object class (7.4.4).

D.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 2863.

D.5.1 Relationship to the Interfaces MIB

IETF RFC 2863, 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 2863 to avoid over constraining the MIB, thereby precluding management of certain media types.

Section 3.3 of IETF RFC 2863 enumerates several areas that a media-specific MIB has to clarify. Each of these areas is addressed in D.5.2 and D.5.3. The implementer is referred to IETF RFC 2863 in order to understand the general intent of these areas.

In IETF RFC 2863, 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 Aggregation 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.

D.5.2 Layering model

This annex assumes the interpretation of the Interfaces Group to be in accordance with IETF RFC 2863, 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).

D.5.3 ifStackTable

If the ifStackTable defined in IETF RFC2863 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.

D.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.

D.6 Definitions for Link Aggregation MIB

In the following MIB definition,[11] should there be any discrepancy between the DESCRIPTION text and the BEHAVIOUR DEFINED AS in the corresponding definition in Clause 7, the definition in Clause 7 shall take precedence.

NOTE 1—There is no requirement for persistency of the objects in the implementations of this MIB module. No guidance is provided for the discontinuity of any counters in the MIB module.

NOTE 2—Many managed objects do not have DEFVAL values. Link Aggregation operation is designed to be functional with any value of these parameters.

[11]MIB definitions are available at http://www.ieee802.org/1/pages/MIBS.html.

```
IEEE8023-LAG-MIB DEFINITIONS ::= BEGIN

-- ------------------------------------------------------------
-- IEEE 802.1AX MIB
-- ------------------------------------------------------------


IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Counter64, Integer32,
            TimeTicks, NOTIFICATION-TYPE
        FROM SNMPv2-SMI
    DisplayString, MacAddress, TEXTUAL-CONVENTION, TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF
    InterfaceIndex
        FROM IF-MIB
    PortList
        FROM Q-BRIDGE-MIB
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    ;


lagMIB MODULE-IDENTITY
    LAST-UPDATED "201412180000Z"
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "  WG-URL: http://grouper.ieee.org/groups/802/1/index.html
         WG-EMail: stds-802-1@ieee.org

          Contact: IEEE 802.1 Working Group Chair
           Postal: C/O IEEE 802.1 Working Group
                   IEEE Standards Association
                   445 Hoes Lane
                   P.O. Box 1331
                   Piscataway
                   NJ 08855-1331
                   USA
          E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
    DESCRIPTION
        "The Link Aggregation module for managing IEEE 802.1AX-REV."
    REVISION "201412180000Z"
    DESCRIPTION
        "The Link Aggregation module for managing IEEE 802.1AX."
    REVISION "201201160000Z"
    DESCRIPTION
        "Updated for IEEE 802.1AXbk"
    REVISION "200706290000Z"
    DESCRIPTION
        "References updated 04 Jun 2007 for IEEE 802.1AX"
    REVISION "200006270000Z"
    DESCRIPTION
        "Original publication IEEE 802.3ad"
```

```
    ::= { iso(1) member-body(2) us(840) 802dot3(10006) snmpmibs(300) 43 }


-- ------------------------------------------------------------
-- Textual Conventions
-- ------------------------------------------------------------


LacpKey ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS       current
    DESCRIPTION
        "The Actor or Partner Key value."
    SYNTAX      Integer32 (0..65535)

LacpState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The Actor and Partner State values from the LACPDU."
    REFERENCE
        "7.3.2.1.20"
    SYNTAX      BITS {
                    lacpActivity(0),
                    lacpTimeout(1),
                    aggregation(2),
                    synchronization(3),
                    collecting(4),
                    distributing(5),
                    defaulted(6),
                    expired(7)
                }

DrcpState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Administrative values of DRCP state."
    SYNTAX      BITS {
                    homeGateway(0),
                    neighborGateway(1),
                    otherGateway(2),
                    ippActivity(3),
                    timeout(4),
                    gatewaySync(5),
                    portSync(6),
                    expired(7)
                }

ChurnState ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The state of the Churn Detection machine."
    SYNTAX      INTEGER {
                    noChurn(1),
                    churn(2),
```

```
                    churnMonitor(3) -- deprecated
}

AggState ::= TEXTUAL-CONVENTION
    STATUS     current
    DESCRIPTION
        "The state of the object entry."
    SYNTAX     INTEGER {
                    up(1),
                    down(2)
                }

DrniConvAdminGatewayList ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1x,"
    STATUS     current
    DESCRIPTION
        "The three elements of the octet string represent the
        three portal system numbers in order of priority with
        highest priority first."
    SYNTAX     OCTET STRING (SIZE (3))

PortalLinkList ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "4d,"
    STATUS     current
    DESCRIPTION
        "Each four octets of the octet string represent an
        ifIndex for an Intra-Port Link. The first ifIndex is
        to Portal System 1, the second ifIndex is to Portal
        System 2 and the third ifIndex is to portal System 3.
        The ifIndex of the current portal system is set to zero."
    SYNTAX     OCTET STRING (SIZE (12))

ServiceIdList ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "4d,"
    STATUS     current
    DESCRIPTION
        "A list which contains, in general, a set of Service IDs
        (8.2.2). If the Service IDs are representing VIDs, only a
        single VID is applicable, while in the case that Service IDs
        are representing I-SIDs, more than one I-SIDs are possible.
        Each four octets represent a Service ID which may either be
        I-SID or VID. An empty set is represented as an octet string
        of size zero."
    SYNTAX     OCTET STRING

-- ----------------------------------------------------------
-- subtrees in the LAG MIB
-- ----------------------------------------------------------

lagMIBNotifications OBJECT IDENTIFIER   ::= { lagMIB 0 }
lagMIBObjects OBJECT IDENTIFIER         ::= { lagMIB 1 }
dot3adAggConformance OBJECT IDENTIFIER  ::= { lagMIB 2 }

dot3adAgg OBJECT IDENTIFIER             ::= { lagMIBObjects 1 }
```

```
dot3adAggPort OBJECT IDENTIFIER         ::= { lagMIBObjects 2 }
dot3adDrni OBJECT IDENTIFIER            ::= { lagMIBObjects 4 }
dot3adIPP OBJECT IDENTIFIER             ::= { lagMIBObjects 5 }

-- ------------------------------------------------------------
-- 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,
        dot3adAggPortTable, dot3adDrniTable and
        dot3adIPPAttributeTable."

::= { 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
        "7.3.1"
    ::= { 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
            Integer32,
        dot3adAggActorSystemID
```

```
            MacAddress,
        dot3adAggAggregateOrIndividual
            TruthValue,
        dot3adAggActorAdminKey
            LacpKey,
        dot3adAggActorOperKey
            LacpKey,
        dot3adAggPartnerSystemID
            MacAddress,
        dot3adAggPartnerSystemPriority
            Integer32,
        dot3adAggPartnerOperKey
            LacpKey,
        dot3adAggCollectorMaxDelay
            Integer32
    }

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. NOTE-The aAggID is represented in the
        SMIv2 MIB as an ifIndex-see D.4.1."
    REFERENCE
        "7.3.1.1.1"
    ::= { 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
        "7.3.1.1.9"
    ::= { dot3adAggEntry 2 }


dot3adAggActorSystemPriority OBJECT-TYPE
    SYNTAX       Integer32 (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
        "7.3.1.1.5"
    ::= { dot3adAggEntry 3 }
```

```
dot3adAggActorSystemID OBJECT-TYPE
    SYNTAX       MacAddress
    MAX-ACCESS   read-write
    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 6, 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 6.7)."
    REFERENCE
        "7.3.1.1.4"
    ::= { 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
        "7.3.1.1.6"
    ::= { 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 6.7.2. This
        is a 16-bit read-write value. The meaning of particular Key
        values is of local significance. For an Aggregator that is
        associated with a Portal, the aAggActorAdminKey has to be
        different for each Portal System. Specifically the two most
```

```
            significant bits are set to aDrniPortalSystemNumber
            (7.4.1.1.7). The lower 14 bits may be any value, have
            to be the same in each Portal System within the same Portal,
                and have a default of zero."
        REFERENCE
            "7.3.1.1.7"
        ::= { 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 6.7.2. This is a 16-bit
            read-only value. The meaning of particular Key values is of
            local significance."
        REFERENCE
            "7.3.1.1.8"
        ::= { 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
            "7.3.1.1.10"
        ::= { dot3adAggEntry 8 }


    dot3adAggPartnerSystemPriority OBJECT-TYPE
        SYNTAX       Integer32 (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
            "7.3.1.1.11"
        ::= { 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
        "7.3.1.1.12"
    ::= { dot3adAggEntry 10 }


dot3adAggCollectorMaxDelay OBJECT-TYPE
    SYNTAX       Integer32 (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 Aggregator Client
        or discarding the frame (see 6.2.3.1.1)."
    REFERENCE
        "7.3.1.1.32"
    ::= { 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
        "7.3.1.1.30"
    ::= { 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
        "7.3.1.1.30"
    ::= { dot3adAggPortListEntry 1 }

-- ---------------------------------------------------------------
-- The Aggregation Extension Table
-- ---------------------------------------------------------------

dot3adAggXTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot3adAggXEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that extends dot3adAggTable."
    REFERENCE
        "7.3.1.1"
    ::= { dot3adAgg 3 }

dot3adAggXEntry OBJECT-TYPE
    SYNTAX      Dot3adAggXEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of extension parameters for the Aggregator
        Configuration Table"
    AUGMENTS { dot3adAggEntry }
    ::= { dot3adAggXTable 1 }

Dot3adAggXEntry ::=
    SEQUENCE {
        dot3adAggDescription
            DisplayString,
        dot3adAggName
            DisplayString,
        dot3adAggAdminState
            AggState,
        dot3adAggOperState
            AggState,
        dot3adAggTimeOfLastOperChange
```

```
            Integer32,
        dot3adAggDataRate
            Integer32,
        dot3adAggOctetsTxOK
            Counter64,
        dot3adAggOctetsRxOK
            Counter64,
        dot3adAggFramesTxOK
            Counter64,
        dot3adAggFramesRxOK
            Counter64,
        dot3adAggMulticastFramesTxOK
            Counter64,
        dot3adAggMulticastFramesRxOK
            Counter64,
        dot3adAggBroadcastFramesTxOK
            Counter64,
        dot3adAggBroadcastFramesRxOK
            Counter64,
        dot3adAggFramesDiscardedOnTx
            Counter64,
        dot3adAggFramesDiscardedOnRx
            Counter64,
        dot3adAggFramesWithTxErrors
            Counter64,
        dot3adAggFramesWithRxErrors
            Counter64,
        dot3adAggUnknownProtocolFrames
            Counter64,
        dot3adAggLinkUpDownNotificationEnable
            Integer32,
        dot3adAggPortAlgorithm
            OCTET STRING,
        dot3adAggPartnerAdminPortAlgorithm
            OCTET STRING,
        dot3adAggPartnerAdminPortConversationListDigest
            OCTET STRING,
        dot3adAggAdminDiscardWrongConversation
            TruthValue,
        dot3adAggPartnerAdminConvServiceMappingDigest
            OCTET STRING
    }

dot3adAggDescription OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "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."
    REFERENCE
```

```
            "7.3.1.1.2"
        ::= { dot3adAggXEntry 1 }

    dot3adAggName OBJECT-TYPE
        SYNTAX       DisplayString
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "A human-readable text string containing a locally significant
            name for the Aggregator. This string is read-write."
        REFERENCE
            "7.3.1.1.3"
        ::= { dot3adAggXEntry 2 }

    dot3adAggAdminState OBJECT-TYPE
        SYNTAX       AggState
        MAX-ACCESS   read-write
        STATUS       current
        DESCRIPTION
            "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."
        REFERENCE
            "7.3.1.1.13"
        ::= { dot3adAggXEntry 3 }

    dot3adAggOperState OBJECT-TYPE
        SYNTAX       AggState
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "This read-only value defines the operational state of the
            Aggregator. An operational state of 'up' indicates that the
            Aggregator is available for use by the Aggregator Client;
            a value of 'down' indicates that the Aggregator is not
            available for use by the Aggregator Client."
        REFERENCE
            "7.3.1.1.14"
        ::= { dot3adAggXEntry 4 }

    dot3adAggTimeOfLastOperChange OBJECT-TYPE
        SYNTAX       Integer32
        MAX-ACCESS   read-only
        STATUS       current
        DESCRIPTION
            "The time at which the interface entered its current
            operational state, in terms of centiseconds since the
```

```
        system was last reset. If the current state was entered
        prior to the last reinitialization of the local network
        management subsystem, then this object contains a value of
        zero. The ifLastChange object in the Interfaces MIB defined
        in IETF RFC 2863 is a suitable object for supplying a value
        for aAggTimeOfLastOperChange. This value is read-only.
            Note - aAggTimeOfLastOperChange was defined in terms of
        the aTimeSinceSystemReset variable of IEEE Std 802.3-2008,
        F.2.1, in earlier versions of this standard.
        aTimeSinceSystemReset and ifLastChange have the same meaning."
    REFERENCE
        "7.3.1.1.15"
    ::= { dot3adAggXEntry 5 }

dot3adAggDataRate OBJECT-TYPE
    SYNTAX       Integer32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current data rate, in bits per second, of the aggregate
        link. The value is calculated as the sum of the data rate of
        each link in the aggregation. This attribute is read-only."
    REFERENCE
        "7.3.1.1.16"
    ::= { dot3adAggXEntry 6 }

dot3adAggOctetsTxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "octets"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "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 (7.3.3.1.7, 7.3.3.1.8, 7.3.3.1.9). However, it
        includes frames discarded by the Frame Distribution function of
        the Aggregator (7.3.1.1.25). This value is read-only."
    REFERENCE
        "7.3.1.1.17"
    ::= { dot3adAggXEntry 7 }

dot3adAggOctetsRxOK OBJECT-TYPE
    SYNTAX       Counter64
    UNITS
        "octets"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "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 (7.3.3.1.2,
            7.3.3.1.3, 7.3.3.1.4), or frames discarded by the Frame
            Collection function of the Aggregator (7.3.1.1.26). This value
            is read-only."
        REFERENCE
            "7.3.1.1.18"
        ::= { dot3adAggXEntry 8 }

    dot3adAggFramesTxOK OBJECT-TYPE
        SYNTAX      Counter64
        UNITS
            "frames"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "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 (7.3.3.1.7,
            7.3.3.1.8, 7.3.3.1.9). However, it includes frames discarded
            by the Frame Distribution function of the Aggregator
            (7.3.1.1.25). This value is read-only."
        REFERENCE
            "7.3.1.1.19"
        ::= { dot3adAggXEntry 9 }

    dot3adAggFramesRxOK OBJECT-TYPE
        SYNTAX      Counter64
        UNITS
            "frames"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "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 (7.3.3.1.2, 7.3.3.1.3, 7.3.3.1.4), or frames
            discarded by the Frame Collection function of the Aggregator
            (7.3.1.1.26). This value is read-only."
        REFERENCE
            "7.3.1.1.20"
        ::= { dot3adAggXEntry 10 }

    dot3adAggMulticastFramesTxOK OBJECT-TYPE
        SYNTAX      Counter64
        UNITS
            "frames"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "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 DA other than the broadcast address.
            The count does not include frames transmitted by the
```