INTERNATIONAL STANDARD

ISO/ IEC/IEEE 8802-1AB

Second edition
2017-07

AMENDMENT 2
2023-07

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

Part 1AB: Station and media access control connectivity discovery

AMENDMENT 2: Support for multiframe protocol data units

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

Partie 1AB: Découverte de connectivité des stations et du contrôle d'accès aux supports

AMENDEMENT 2: Support pour les unités de données de protocole multitrame

Reference number
ISO/IEC/IEEE 8802-1AB:2017/Amd.2:2023(E)

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

Email: stds.ipr@ieee.org
Website: www.ieee.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO/IEC documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

ISO/IEC/IEEE 8802-1AB:2017/Amd.2 was prepared by the LAN/MAN of the IEEE Computer Society (as IEEE 802.1ABdh-2021) and drafted in accordance with its editorial rules. It was adopted, under the "fast-track procedure" defined in the Partner Standards Development Organization cooperation agreement between ISO and IEEE, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*.

A list of all parts in the ISO/IEC/IEEE 8802 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

IEEE Std 802.1ABdh™-2021
(Amendment to IEEE Std 802.1AB™-2016
as amended by IEEE Std 802.1ABcu™-2021)

IEEE Standard for Local and metropolitan area networks—

Station and Media Access Control Connectivity Discovery

Amendment 2: Support for Multiframe Protocol Data Units

Developed by the

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 8 December 2021

IEEE SA Standards Board

Abstract: This amendment to IEEE Std 802.1AB™-2016 specifies protocols, procedures, and managed objects that support the transmission and reception of a set of Link Layer Discovery Protocol (LLDP) Type/Length/Values (TLVs) that exceed the space available in a single frame.

Keywords: IEEE 802®, IEEE 802.1AB™, IEEE 802.1ABdh™, link layer discovery protocol, management information base, multiframe LLDPDUs, topology discovery, topology information

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

PDF: ISBN 978-1-5044-8301-8 STD25173
Print: ISBN 978-1-5044-8302-5 STDPD25173

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (https://standards.ieee.org/ipr/disclaimers.html), appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents."

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

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA, 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 makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. 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: THE NEED TO PROCURE 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 is 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, nor 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 the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, 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 evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the IEEE SA myProject system.[1] An IEEE Account is needed to access the application.

Comments on standards should be submitted using the Contact Us form.[2]

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 constitute 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.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

[1]Available at: https://development.standards.ieee.org/myproject-web/public/view.html#landing.

[2]Available at: https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US 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 licensing fees, 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; https://www.copyright.com/. 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 10 years. When a document is more than 10 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 IEEE Xplore or contact IEEE.[3] For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website.[4] Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore. Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the IEEE SA Patent Policy.[5]

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

[3] Available at: https://ieeexplore.ieee.org/browse/standards/collection/ieee.

[4] Available at: https://standards.ieee.org/standard/index.html.

[5] Available at: https://standards.ieee.org/about/sasb/patcom/materials.html.

filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://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.

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Participants

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

Glenn Parsons, *Chair*
Jessy Rouyer, *Vice-Chair*
János Farkas, *Chair, Time-Sensitive Networking Task Group*
Craig Gunther, *Vice-Chair, Time-Sensitive Networking Task Group*
Stephen Haddock, *Editor*

Astrit Ademaj
Ralf Assmann
Venkat Arunarthi
Rudy Belliardi
Christian Boiger
Paul Bottorff
Radhakrishna Canchi
Feng Chen
Abhijit Choudhury
Paul Congdon
Rodney Cummings
Josef Dorr
Hesham Elbakoury
Anna Engelmann
Thomas Enzinger
Donald Fedyk
Norman Finn
Geoffrey Garner
Amrit Gopal
Marina Gutierrez
Mark Hantel
Jerome Henry
Marc Holness
Daniel Hopf
Woojung Huh
Satoko Itaya
Yoshihiro Ito
Michael Karl
Stephan Kehrer
Marcel Kiessling
Randy Kelsey
Gavin Lai
James Lawlis
Joao Lopes
Lily Lv
Christophe Mangin
Scott Mansfield
Kenichi Maruhashi
Olaf Mater
David McCall
Larry McMillan
John Messenger
Hiroki Nakano
Hiroshi Ohue
Donald R. Pannell
Razvan Pere
Michael Potts
Dieter Proell
Wei Qiu
Karen Randall
Maximilian Riegel
Silvana Rodrigues
Atsushi Sato
Frank Schewe
Michael Seaman
Maik Seewald
Ramesh Sivakolundu
Johannes Specht
Manus Stanica
Guenter Steindl
Liyang Sun
Karim Traore
Max Turner
Balazs Varga
Ganesh Venkatesan
Tongtong Wang
Xinyuan Wang
Karl Weber
Leon Wessels
Ludwig Winkel
Jordon Woods
Takahiro Yamaura
Yue Yin
Uwe Zeier
Nader Zein
William Zhao
Helge Zinner

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

Thomas Alexander
Harry Bims
Christian Boiger
Vern Brethour
William Byrd
Paul Cardinal
Pin Chang
Diego Chiozzi
János Farkas
Avraham Freedman
Craig Gunther
Stephen Haddock
Marco Hernandez
Werner Hoelzl
Oliver Holland
Pranav Jha
Lokesh Kabra
Piotr Karocki
Stephan Kehrer
Randy Kelsey
Stuart Kerry
Evgeny Khorov
Yongbum Kim
Hyeong Ho Lee
Ting Li
Christophe Mangin
Scott Mansfield
Jonathon Mclendon
Satoshi Obara
Glenn Parsons
Bansi Patel
Arumugam Paventhan
Clinton Powell
Dieter Proell
Alon Regev
Maximilian Riegel
Jessy Rouyer
Frank Schewe
Michael Seaman
Eugene Stoudenmire
Walter Struppler
Mitsutoshi Sugawara
Bo Sun
Max Turner
John Vergis
Stephen Webb
Karl Weber
Scott Willy
Yu Yuan
Oren Yuen

When the IEEE SA Standards Board approved this standard on 8 December 2021, it had the following membership:

Gary Hoffman, *Chair*
Jon Walter Rosdahl, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Edward A. Addy	Howard Li	Mehmet Ulema
Doug Edwards	Daozhuang Lin	Lei Wang
Ramy Ahmed Fathy	Kevin Lu	F. Keith Waters
J. Travis Griffith	Daleep C. Mohla	Karl Weber
Thomas Koshy	Chenhui Niu	Sha Wei
Joseph L. Koepfinger*	Damir Novosel	Howard Wolfman
David J. Law	Annette Reilly	Daidi Zhong
	Dorothy Stanley	

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1ABdh-2021, IEEE Standard for Local and metropolitan area networks—Station and Media Access Control Connectivity Discovery—Amendment 2: Support for Multiframe Protocol Data Units.

The first edition of IEEE Std 802.1AB™ was published in 2005, and revisions were published in 2009 and 2016. An amendment, IEEE Std 802.1ABcu™-2021, specifies a Unified Modeling Language (UML)-based information model and a YANG data model. This amendment, IEEE Std 802.1ABdh™-2021, specifies protocols, procedures, and managed objects that support the transmission and reception of a set of Link Layer Discovery Protocol (LLDP) Type/Length/Values (TLVs) that exceed the space available in a single frame.

Contents

IEEE Standard for Local and metropolitan area networks—

Station and Media Access Control Connectivity Discovery

Amendment 2: Support for Multiframe Protocol Data Units

(This amendment is based on IEEE Std 802.1AB™-2016 as amended by IEEE Std 802.1ABcu™-2021.)

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

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Deletions and insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.[6]

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

3. Definitions and numerical representation

3.1 Definitions

Insert the following new definitions in 3.1 in alphanumerical order:

Extended Link Layer Discovery Protocol (XLLDP): An LLDP extension used to retrieve the frames beyond the Normal LLDPDU of a multi-frame LLDP database.

Extension LLDPDU (XPDU): An LLDPDU containing an Extension Identifier TLV and not containing a Time to Live TLV or Extension Request TLV.

Extension Request LLDPDU (XREQ): An LLDPDU containing an Extension Request TLV and not containing a Time to Live TLV or Extension Identifier TLV.

Manifest LLDPDU: A Normal LLDPDU containing a Manifest TLV.

Normal LLDPDU: An LLDPDU containing a Time To Live TLV and not containing an Extension Request TLV or an Extension Identifier TLV.

Scope MAC Address: The MAC address used as the destination address in Normal LLDPDUs, which determines the limits of propagation in the network.

shutdown LLDPDU: A Normal LLDPDU with a TTL value of zero.

4. Acronyms and abbreviations

Insert the following new abbreviations in Clause 4 in alphabetical order:

XLLDP	Extended Link Layer Discovery Protocol
XPDU	Extension LLDPDU
XREQ	Extension Request LLDPDU

5. Conformance

5.3 Required capabilities

Change list items j), k), and l) in the lettered list in 5.3 as follows:

j) If receipt of LLDPDUs is supported, for every set of TLVs (the basic management set, extended LLDP set, and any organizationally specific sets) supported, support shall be implemented for receipt of every TLV defined in the set.

k) If transmission of LLDPDUs is supported, for every set of TLVs (the basic management set, extended LLDP set, and any organizationally specific sets) supported, support shall be implemented for transmission of every TLV defined in the set.

l) If transmission of LLDPDUs is supported, for every set of TLVs (the basic management set, extended LLDP set, and any organizationally specific sets) supported, a capability shall be implemented for users to determine which optional TLVs are transmitted in any particular LLDPDU.

Insert new item p) at the end of the lettered list in 5.3 (as modified by IEEE Std 802.1ABcu-2021) as follows:

p) If the Extended Link Layer Discovery Protocol is supported, then
   1) The extended LLDP set of TLVs shall be implemented as defined in 8.5a.
   2) The Extension Request and Extension LLDPDU formats (8.2) and addressing (7.1.2) shall be implemented.
   3) The extended receive state machine (Figure 9-2a) shall be implemented.

6. Principles of operation

Change the first sentence of the second paragraph of the introduction to Clause 6 as follows:

This clause describes general principles of LLDP and Extended LLDP (XLLDP) operation.

6.1 Transmission and reception

Change 6.1 as follows:

The information fields in each LLDP frame are contained in a Link Layer Discovery Protocol Data Unit (LLDPDU) as a sequence of variable length information elements, that each include type, length, and value fields (known as TLVs), where

— ~~a)~~ Type identifies what kind of information is being sent.
— ~~b)~~ Length indicates the length of the information string in octets.
— ~~c)~~ Value is the actual information that needs to be sent (for example, a binary bit map or an alpha-numeric string that can contain one or more fields).

~~Each LLDPDU contains the following three mandatory TLVs (see Table 8-1), and can contain optional TLVs as selected by network management:~~ Each LLDPDU begins with a mandatory sequence of three TLVs, as specified in 8.2, and can contain zero or more additional optional TLVs as selected by network management. The first three TLVs are:

1) ~~d)~~ A Chassis ID TLV.
2) ~~e)~~ A Port ID TLV.
3) ~~f)~~ A Time To Live TLV or, if XLLDP is supported, an Extension Request TLV or an Extension Identifier TLV.

~~g) Zero or more optional TLVs, as allowed by the maximum size of the LLDPDU.~~
~~h) An optional End Of LLDPDU TLV.~~

The first two TLVs contain the chassis ID and the port ID values. These are concatenated to form a logical MSAP identifier. The combination of the MSAP identifier and the Scope MAC Address identify the ~~that is used by the recipient to identify the sending~~ LLDP agent/port that is the source of the information in the TLVs. Both the chassis ID and port ID values can be defined in a number of convenient forms. Once selected, however, the chassis ID/port ID value combination remains the same as long as the particular port remains operable (8.5.2, 8.5.3, 9.2.4.1).

~~NOTE 1—The statement above is true for any LLDP agent; however, there can be multiple LLDP agents sending and receiving LLDPDUs using different MAC addresses.~~

~~A non-zero value in the time to live (TTL) field of the Time To Live TLV tells the receiving LLDP agent how long all information pertaining to this LLDPDU's MSAP identifier is valid so that all the associated information can later be automatically discarded by the receiving LLDP agent if the sender fails to update it in a timely manner. A zero value indicates that any information pertaining to this LLDPDU's MSAP identifier is to be discarded immediately.~~

The third TLV differentiates between three types of LLDPDUs:

a) A Normal LLDPDU, where the third TLV is a Time To Live TLV that tells the receiving LLDP agent how long all information pertaining to this LLDPDU's MSAP identifier is valid so that all of the associated information can later be automatically discarded by the receiving LLDP agent if the

sender fails to update it in a timely manner. It is useful to further differentiate two special cases of a Normal LLDPDU:

1) A shutdown LLDPDU, where the third TLV is a Time To Live TLV with a TTL value of zero indicating that any information pertaining to this LLDPDU's MSAP identifier is to be discarded immediately. A TTL value of zero can be used, for example, to signal that the sending port has initiated a port shutdown procedure.

2) A Manifest LLDPDU, where the third TLV is a Time To Live TLV with a non-zero TTL value and one of the subsequent TLVs is a Manifest TLV.

b) An Extension Request LLDPDU (XREQ), where the third TLV is an Extension Request TLV that the recipient of a Normal LLDPDU containing a Manifest TLV uses to request one or more of the Extension LLDPDUs described in the Manifest TLV. The Extension Request LLDPDU does not contain a Time to Live TLV.

c) An Extension LLDPDU (XPDU), where the third TLV is an Extension Identifier TLV. The Extension LLDPDU does not contain a Time to Live TLV.

~~NOTE 2—A TTL value of zero can be used, for example, to signal that the sending port has initiated a port shutdown procedure.~~

NOTE —The receive state machine (in 9.2.9 and in prior versions of this standard) specifies that an implementation not supporting XLLDP will discard any received LLDPDUs not containing a TTL TLV as the third TLV (i.e., will discard any received Extension Request or Extension LLDPDUs). The state machine further specifies that only implementations supporting XLLDP will process a Manifest TLV and subsequently transmit Extension Request LLDPDUs, and that only implementations supporting XLLDP will process Extension Request TLVs and subsequently transmit Extension LLDPDUs.

~~The~~ An optional End Of LLDPDU TLV marks the end of the LLDPDU.

The format for the LLDPDU is defined in 8.2. The TLV categories and the basic TLV format are defined in 8.3 and 8.4. The specific format and field contents for the Chassis ID TLV are defined in 8.5.2; for the Port ID TLV, in 8.5.3; for the Time To Live TLV, in 8.5.4; and for the End Of LLDPDU TLV, in 8.5.1.

6.2 LLDP operational modes

Change 6.2 as follows:

~~LLDP is a one way protocol.~~ The information advertised by LLDP is transferred in a single direction. An LLDP agent can ~~transmit~~ periodically advertise information about the capabilities and current status of the system associated with its MSAP identifier. The LLDP agent can also ~~receive~~ collect information about the capabilities and current status of the system associated with a remote MSAP identifier. LLDP does not itself contain a mechanism for ~~soliciting~~ the LLDP agent to solicit specific information from other LLDP agents beyond what is being advertised by those LLDP agents, nor does it provide a specific means of confirming the receipt of information. Extended LLDP allows an LLDP agent collecting information to request detailed information (via an Extension Request LLDPDU) that is only advertised in a summary form (in a Manifest TLV), however this does not change the inherent LLDP property that the choice of information provided is at the discretion of the advertising LLDP agent.

NOTE—The LLDP protocol is designed to advertise information useful for discovering pertinent information about a remote port and to populate topology MIBs. It is not intended to act as a configuration protocol for remote systems, nor as a mechanism to signal control information between ports. During the operation of LLDP, it may be possible to discover configuration inconsistencies between systems on the same IEEE 802 LAN. LLDP does not provide a mechanism to resolve those inconsistencies. Rather, it provides a means to report discovered information to higher layer management entities.

LLDP allows the ~~transmitter and the receiver~~ advertising function and collecting function of the LLDP agent to be separately enabled, making it possible to configure an implementation so the local LLDP agent can

either ~~transmit~~ advertise only or ~~receive~~ collect only, or so the local LLDP agent can ~~transmit~~ advertise and ~~receive~~ collect LLDP information.

6.5 Transmission principles

Change 6.5 as follows:

A transmission cycle ~~Transmission~~ can be initiated either by the expiration of a transmit countdown timing counter or by a change in the value of one or more of the information elements (managed objects) associated with the local system. When a transmit cycle is initiated, the LLDP management entity extracts the managed objects from the LLDP local system MIB and formats this information into TLVs. The TLVs are inserted into an LLDPDU that is passed to the LLDP transmit module. When Extended LLDP is supported, some of the TLVs are inserted into one or more Extension LLDPDUs that are passed to the LLDP transmit module when requested by the receipt of a Extension Request LLDPDU. The LLDP transmit module prepends addressing parameters to the LLDPDU as defined in 8.2, 8.3, and 8.4. The LLDPDU and TLV formats are defined in Clause 8. The LLDP transmit state machine is described in 9.2.1 and 9.2.8.

NOTE 1—Because a transmission cycle can be initiated whenever a change occurs within the LLDP local system MIB, it is possible that a series of successive changes over a short period of time could trigger a number of LLDP frames to be sent, each reporting only a single change. LLDP utilizes a transmission delay timer that can be set by network management to limit the ~~ensure that there is a defined maximum~~ number of LLDP transmission cycles ~~LLDPDU transmissions~~ in a given time period.

NOTE 2—Under normal circumstances, the information in the receiving LLDP agent's remote systems MIB is refreshed periodically to avoid being discarded due to ageing. To prevent the receiving LLDP agent's remote systems MIB information being aged out because a refresh frame has been lost in transmission, the sending LLDP agent typically sets the TTL value so that several refresh cycles can occur before the received MIB information ages out.

LLDP can disclose information about a device and network that could be considered sensitive in certain environments. Implementers are encouraged to provide controls that take into account the link protection characteristics when including information in an LLDP message. Different information can be included if an LLDP message is being sent on the uncontrolled port, the controlled port, or a MACsec protected connectivity association.

6.6 Reception principles

6.6.1 LLDPDU and TLV error handling

Change the first paragraph of 6.6.1 as follows:

The LLDPDU is checked to verify ~~ensure~~ that it contains the correct sequence of three mandatory TLVs at the beginning of the frame, as specified in 8.2, ~~(Chassis ID TLV, Port ID TLV, and Time To Live TLV)~~ and then each optional TLV is validated in succession. LLDPDUs that contain detectable errors in the first three mandatory TLVs are discarded. Optional TLVs that contain detectable errors are discarded [see 9.2.7.7.2 c)]. TLVs that are not recognized, but that also contain no basic format errors, are assumed to be valid and are stored for possible later retrieval by network management (see 9.2.7.7.1 and 9.2.7.4). If the End Of LLDPDU TLV is present, any octets that follow it are discarded.

6.7 Systems with multiple LLDP Agents

Change the first paragraph of 6.7 as follows:

Each LLDP agent advertises a single set of information in the various TLVs it encodes in each transmission cycle ~~transmitted LLPDU~~, and is associated with the MSAP that supports the LLC entity that the LLDP agent uses to transmit and receive. Each LLDP agent uses its LSAP directly, without the use of any additional multiplexing or addressing above the LSAP to support the use of that LSAP by multiple LLDP agents; and each LLC entity provides service to one and only one protocol entity at each of its LSAPs that it supports, using the service provided by a single MSAP. It follows that each LLDP agent makes use of a unique MSAP, and that the LLDP agent can be uniquely identified by the receiving agent using the MSAP's identifier and the MAC address that determines the LLDP agent's address scope, as specified in ~~6.2~~ 6.1. A single LLDP management entity can support the operation of multiple LLDP agents within the same system. Figure 6-2 illustrates the relationship between the LLDP agents, LLC Entities, MSAPs, and the LLDP management entity.

Change the second sentence of the second paragraph of 6.7 as follows:

These sets of peers can be determined by the way that the MAC service is supported within the system (see IEEE Std 802.1AC~~-2015~~ for a description of the connectionless connectivity that characterizes an IEEE 802 LAN).

Change the final paragraph of 6.7 as follows (Figure 6-7 remains unchanged):

More than one LLDP agent, each using a different address scope, can be instantiated for a given system port by adding a simple shim that provides the necessary distinct MSAPs by multiplexing and demultiplexing between those MSAPs and a common MSAP for the port, as illustrated in Figure 6-7. Each MSAP shown in Figure 6-7 is allowed to shared the same MSAP Identifier and individual MAC address. The LLDP agents are differentiated by address scope used, i.e., by the MAC address used as a Destination Address in frames containing a Normal LLDPDU.

Insert new subclause 6.9 after 6.8 as follows:

6.9 Extended LLDP (XLLDP)

Extended LLDP allows an LLDP agent to advertise more TLVs than would fit within a single LLDPDU by transmitting the additional TLVs in one or more Extension LLDPDUs. The selection of TLVs to be included in LLDPDUs is controlled by management, but the mapping of TLVs to the initial Normal LLDPDU or to an Extension LLDPDU is implementation dependent (10.2.2). When a transmission cycle is initiated, a unique description of each Extension LLDPDU is encoded in a Manifest TLV, which is then included in the Normal LLDPDU transmitted at the beginning of the cycle. A Normal LLDPDU containing a Manifest TLV is referred to as a Manifest LLDPDU. Changes to any of the TLVs in an Extension LLDPDU changes the description included in the Manifest TLV, and thus triggers a new transmission cycle.

An implementation not supporting XLLDP that receives a Manifest TLV treats it as an unrecognized TLV and only stores the TLVs that were included in the Normal LLDPDU. An implementation supporting XLLDP that receives a Manifest TLV then requests the transmission of any new or modified Extension LLDPDUs by sending an Extension Request LLDPDU to the LLDP agent sourcing the Manifest LLDPDU. One or more Extension LLDPDUs can be requested in a single Extension Request LLDPDU. A new Extension Request LLDPDU can be transmitted when all Extension LLDPDUs previously requested have been received. This allows the requesting LLDP agent to pace the rate of information transfer. A timer on the request allows an Extension Request LLDPDU to be retried if it appears that either the request or a response has been lost.

Extension Request LLDPDUs and Extension LLDPDUs are transmitted with an individual MAC address as the destination address. The destination address in an Extension Request LLDPDU is taken from the contents of the Return MAC Address field of the received Manifest TLV. The requested Extension LLDPDUs are transmitted with a destination address taken from the contents the Return MAC Address field of the Extension Request TLV.

7. LLDPDU transmission, reception, and addressing

Change the note in the introduction to Clause 7 as follows:

NOTE—For the purposes of this standard, the terms "LLC" and "LLC entity" include the service provided by the operation of entities that support protocol discrimination using an EtherType, i.e., ~~Ethertype Protocol Discrimination (EPD) as specified in IEEE Std 802.~~ protocol discrimination based on the Type interpretation of the Length/Type field as specified in IEEE Std 802.3, or the use of the SNAP encapsulation of EtherTypes as described in IEEE Std 802. Hence, "LSAP" in the above description can refer to the use of ISO/IEC 8802-2 [B15] LLC addresses, or an EtherType, as a means of higher layer protocol identification.

7.1 Destination address

Insert a new heading 7.1.1 at the start of 7.1 and change the first paragraph of what was 7.1 as follows:

7.1.1 Destination address for Normal LLDPDUs

The MAC address used as the destination address in Normal LLDPDUs is referred to as the Scope MAC Address because it determines the limits of propagation of the LLDPDU within a network. A set of standard group MAC addresses that can be used by LLDP as the Scope MAC Address is specified in Table 7-1. These addresses are in the range of IEEE Std 802.1Q C-VLAN and MAC Bridge component Reserved addresses (see Table 8-1 in IEEE Std 802.1Q-2014), the S-VLAN component Reserved addresses (see Table 8-2 in IEEE Std 802.1Q-2014), and the TPMR component Reserved addresses (see Table 8-3 in IEEE Std 802.1Q-2014). The choice of address used determines the scope of propagation of LLDPDUs within a bridged LAN, as follows:

Change the last two paragraphs of what was 7.1, the heading for Table 7-2, and NOTE 7 as follows:

In addition to the prescribed support for standard group MAC addresses shown in Table 7-1, implementations of LLDP may support the following as the Scope MAC Address ~~destination addresses for LLDPDUs~~:

d) Any group MAC address.
e) Any individual MAC address.

Support for the use of each of these destination addresses, for both transmission and reception of Normal LLDPDUs, is either mandatory, recommended, permitted, or not permitted, according to the type of system in which LLDP is implemented, as shown in Table 7-2.

Table 7-2—Support for the Scope MAC Address ~~addresses~~ in different systems

Address	C-VLAN Bridge	S-VLAN Bridge	TPMR Bridge	End station
Nearest bridge	Mandatory	Mandatory	Mandatory	Mandatory
Nearest non-TPMR bridge	Mandatory	Mandatory	Not permitted	Recommended
Nearest Customer Bridge	Mandatory	Not permitted	Not permitted	Recommended
Any other group MAC address	Permitted	Permitted	Permitted	Permitted
Any individual MAC address	Permitted	Permitted	Permitted	Permitted

NOTE 7—If an individual MAC address is specified as the Scope MAC Address, it is used as the destination for transmitted LLDPDUs and also used to validate received LLDPDUs. The primary purpose of using an individual address as the Scope MAC Address is that an LLDP agent configured as "transmit only" sends Normal LLDPDUs to a specific receiving LLDP agent. It is also allowable for an LLDP agent configured as "receive only" to use an individual address (typically the individual MAC address of the LLDP agent's MSAP) so that only information from an LLDP agent configured as "transmit only" using this same address is collected. Using an individual MAC address as the destination address for an LLDP agent configured to "transmit and receive" is allowed, although it would not typically be useful. ~~If an individual MAC address is used as the destination address by a given LLDP Agent, then that agent would not also receive LLDPDUs on that address; it makes no sense for an LLDP Agent to send LLDP frames to an individual MAC address that it also receives LLDP frames on. Similarly, if an Agent is able to receive LLDPDUs sent to a given individual MAC address, it would not also transmit LLDPDUs to that address. Therefore, Agents associated with individual MAC addresses are either transmitters or receivers of LLDPDUs for a given address, but not both.~~

Insert new subclause 7.1.2 after NOTE 8 in what has become 7.1.1 as follows:

7.1.2 Destination address for Extension Request and Extension LLDPDUs

The destination address in an Extension Request LLDPDU or an Extension LLDPDU is an individual address. Extension Request LLDPDUs are sent in response to the receipt of a Manifest LLDPDU, and use the contents of the Return MAC Address field of the Manifest TLV as the destination address of an Extension Request LLDPDU. Likewise, Extension LLDPDUs are sent in response to the receipt of an Extension Request LLDPDU, and use the contents of the Return MAC Address field of the Extension Request TLV as the destination address of the Extension LLDPDU.

7.2 Source address

Change 7.2 as follows:

The source address shall be the individual MAC address of the MSAP supporting the LLDP agent ~~sending station or port~~.

7.4 LLDPDU reception

Change list item a) in 7.4 as follows:

a) The destination MAC address is equal to the individual MAC address of the MSAP supporting the LLDP agent, or the Scope MAC Address ~~address~~ associated with the corresponding LLDP agent ~~Agent~~; and

8. LLDPDU and TLV formats

8.2 LLDPDU format

Change the first paragraph and the lettered list in 8.2 as follows:

The LLDPDU shall contain the following ordered sequence of three mandatory TLVs followed by zero or more optional TLVs as shown in Figure 8-1. An End of LLDPDU TLV may be present as the last TLV in the LLDPDU.

a) Three mandatory TLVs shall be included at the beginning of each LLDPDU and shall be in the order shown.
   1) Chassis ID TLV
   2) Port ID TLV
   3) Time To Live TLV or, if XLLDP is supported, Extension Request TLV or Extension Identifier TLV

b) Optional TLVs as selected by network management (may be inserted in any order).

NOTE 1—"Optional" in the sense that they are not required for LLDP operation; however, their presence could be required by other system elements that use LLDP.

c) If the End Of LLDPDU TLV is present, it shall be the last TLV in the LLDPDU.

Replace Figure 8-1 with the following figure:

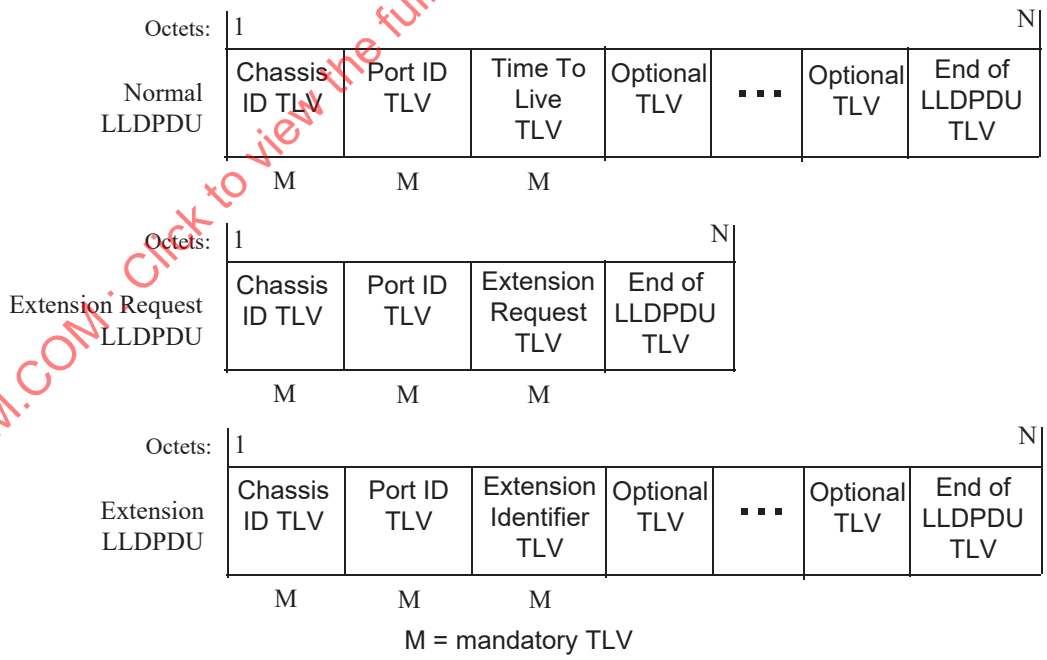

Figure 8-1—LLDPDU formats

Change the last paragraph and NOTE 2 in 8.2 as follows:

The maximum length of the LLDPDU ~~shall be~~ is less than or equal to the maximum information field length allowed by the particular transmission rate and protocol. In IEEE 802.3 MACs, for example, the maximum LLDPDU length is the maximum data field length for the basic, untagged MAC frame (1500 octets).

NOTE 2—Implementations can restrict the maximum length of the LLDPDU to a value smaller than what would be allowed by the underlying MAC technology. For example, an end station or bridge implementing Time-Sensitive Networking (TSN) features could restrict the maximum length to limit delay variation when transmitting frames in time sensitive streams.

NOTE ~~2~~3—There is no defined minimum length of an LLDPDU, other than that implied by the requirement that conformant implementations support the mandatory TLVs specified in Table 8-1.

8.3 TLV categories

Change 8.3 as follows:

The general format of LLDP TLVs is defined in 8.4. The TLVs are grouped into ~~two general~~ three categories as follows:

a) A set of TLVs that are considered to be basic to the management of network stations. ~~and that are~~ Support of the basic management set is a required capability of all LLDP implementations. Each TLV in this category is identified by a unique TLV type value that indicates the particular kind of information contained in the TLV. The specific definitions and usage rules for the basic management set TLVs are defined in 8.5.

b) A set of TLVs that support Extended LLDP (XLLDP) operation. Support of the extended LLDP set is a required capability of all LLDP implementations supporting XLLDP. Each TLV in this category is identified by a unique TLV type value that indicates the particular kind of information contained in the TLV. The specific definitions and usage rules for the extended LLDP set TLVs are defined in 8.5a.

c) ~~b)~~ Organizationally specific extension sets of TLVs that are defined by standards groups such as IEEE 802.1 and IEEE 802.3 and others to enhance management of network stations that are operating with particular media and/or protocols. The format of Organizationally Specific TLVs, along with field definitions and usage rules common to all Organizationally Specific TLVs, are defined in 8.6.
    1) TLVs in this category are identified by a common TLV type value that indicates the TLV as belonging to the set of Organizationally Specific TLVs.
    2) Each organization is identified by its organizationally unique identifier, consisting of either an Organizationally Unique Identifier (OUI) or a Company ID (CID)[7].
    3) Organizationally Specific TLV subtype values indicate the kind of information contained in the TLV.

~~The basic TLV format and general field definition rules are defined in 8.4. Specific definitions and usage requirements for all basic management set TLVs are defined in 8.5. Usage rules/requirements that pertain to each individual basic TLV are contained in the definition for that particular TLV.~~

~~The basic format and the field definition/general usage rules/restrictions for Organizationally Specific TLVs are defined in 8.6. Usage rules/requirements that pertain to each individual basic TLV are contained in the definition for that particular TLV.~~

[7] Organizationally Unique Identifier and Company ID assignments are administered by the IEEE Registration Authority, and Organizationally Unique Identifier assignments are included in MAC Address-Large (MA-L) assignments; see http://standards.ieee.org/regauth.

8.4 Basic TLV format

8.4.1 TLV type

Change Table 8-1 as follows:

Table 8-1—TLV type values

TLV type [a]	TLV name	Usage in LLDPDU	Reference
0	End Of LLDPDU	Optional	8.5.1
1	Chassis ID	Mandatory	8.5.2
2	Port ID	Mandatory	8.5.3
3	Time To Live	Mandatory	8.5.4
4	Port Description	Optional	8.5a
5	System Name	Optional	8.5.6
6	System Description	Optional	8.5.7
7	System Capabilities	Optional	8.5.8
8	Management Address	Optional	8.5.9
9	Manifest	Optional	8.5a.1
10	Extension Request	Optional	8.5a.2
11	Extension Identifier	Optional	8.5a.3
~~9~~12–126	Reserved for future standardization	—	—
127	Organizationally Specific TLVs	Optional	—

[a]TLVs with type values 0–8 are members of the basic management set. TLVs with type values 9–11 are members of the extended LLDP set.

8.5 Basic management TLV set formats and definitions

8.5.2 Chassis ID TLV

Change the first paragraph of 8.5.2 as follows:

The Chassis ID TLV is a mandatory TLV that identifies the chassis containing the IEEE 802 LAN station associated with the MSAP Identifier associated with the LLDP agent containing the LLDP local system MIB that is the source of the information in TLVs ~~transmitting LLDP agent~~. There are several ways in which a chassis may be identified and a chassis ID subtype is used to indicate the type of component being referenced by the chassis ID field. Each LLDPDU shall contain one, and only one, Chassis ID TLV and the chassis ID field value shall remain constant for all LLDPDUs while the MAC status parameter for the Port remains operational.

8.5.3 Port ID TLV

Change the first paragraph of 8.5.3 as follows:

The Port ID TLV is a mandatory TLV that identifies the port component of the MSAP identifier associated with the MSAP Identifier associated with the LLDP agent containing the LLDP local system MIB that is the source of the information in TLVs ~~transmitting LLDP agent~~. As with the chassis, there are several ways in which a port may be identified. A port ID subtype is used to indicate how the port is being referenced in the port ID field. Each LLDPDU shall contain one, and only one, Port ID TLV. The port ID value shall remain constant for all LLDPDUs while the transmitting port remains operational.

8.5.4 Time To Live TLV

Change the second paragraph of 8.5.4 as follows:

The Time To Live TLV is mandatory for all Normal LLDPDUs, and shall be the third TLV in the LLDPDU. Its format is shown in Figure 8-6.

8.5.4.2 Time To Live TLV usage rules

Change 8.5.4.2 as follows:

A Normal ~~An~~ LLDPDU shall contain exactly one Time To Live TLV as the third TLV, and shall not contain more than one Time To Live TLV. A Time To Live TLV shall not be contained in an Extension LLDPDU or an Extension Request LLDPDU.

Insert new subclause 8.5a after 8.5 as follows:

8.5a Extended LLDP set TLV formats and definitions

8.5a.1 Manifest TLV

The Manifest TLV allows an LLDP agent supporting XLLDP to advertise more TLVs than fits in a single LLDPDU. The Manifest TLV contains a list of Extension PDUs that convey the additional TLVs. The format for this TLV is shown in Figure 8-11a.

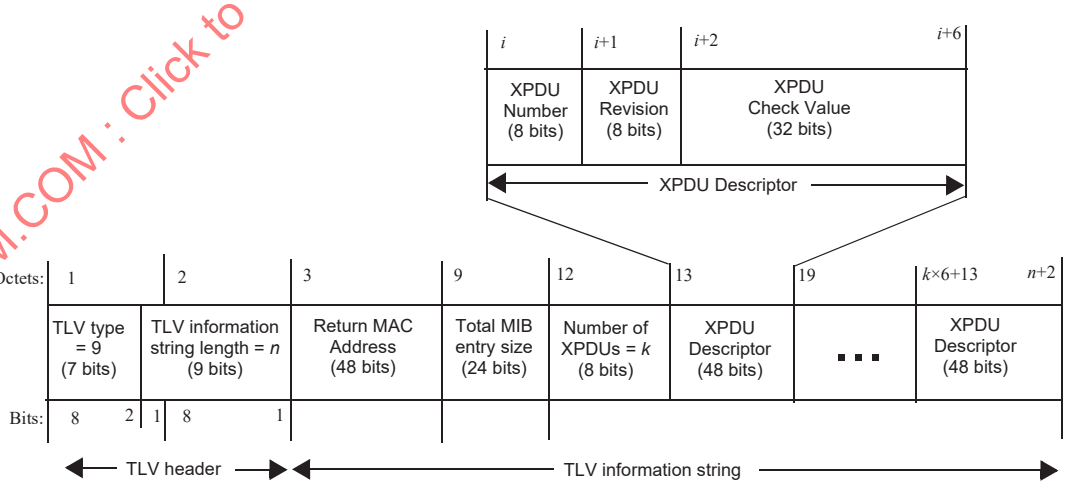

Figure 8-11a—Manifest TLV format

8.5a.1.1 TLV information string length

The TLV information string length field shall contain the number of octets in the TLV information string.

8.5a.1.2 Return MAC Address

The individual MAC Address of the MSAP supporting the LLDP agent, to be used as the destination address for Extension Request LLDPDUs generated in response to this Manifest TLV.

8.5a.1.3 Total MIB entry size

The total MIB entry size is the number of octets in all TLVs associated with this MSAP Identifier (regardless of the number of XPDUs used to convey those TLVs). It is the sum of the information string length fields in all TLVs, plus two times the number of TLVs, and includes the TLVs in the initial Normal LLDPDU. This allows the LLDP remote systems MIB manager to determine whether it expects to have space to store the information prior to requesting any XPDUs.

8.5a.1.4 Number of XPDUs

The number of XPDUs field contains the count of the XPDU Descriptors in this TLV. Since the maximum size of the TLV information string is 511 octets, the minimum number of XPDU Descriptors is 1 and the maximum number is 83. When the number of XPDU Descriptors is k, the TLV information string length is at least $k \times 6 + 10$, but it need not be exactly $k \times 6 + 10$. This allows an implementation to use a fixed length TLV information string with a variable number of XPDU Descriptors.

8.5a.1.5 XPDU Descriptor

An XPDU Descriptor comprises an 8-bit XPDU Number, an 8-bit XPDU Revision, and a 32-bit XPDU Check Value. The XPDU Number uniquely identifies each XPDU in a manifest. The XPDU Revision is assigned a random value at initialization, and is incremented (modulo 256) each time there is a change to one or more TLVs in the XPDU. The XPDU Check Value is the low order 32 bits of an MD5 hash calculated across all octets of the XPDU.

8.5a.1.6 Manifest TLV usage rules

An LLDPDU shall not contain more than one Manifest TLV. Each XPDU Descriptor associated with an MSAP identifier and Scope MAC Address shall have a unique XPDU Number.

8.5a.2 Extension Request TLV

The Extension Request TLV allows an LLDP agent supporting XLLDP to request that the neighbor transmit specific Extension PDUs. The format for this TLV is shown in Figure 8-11b.

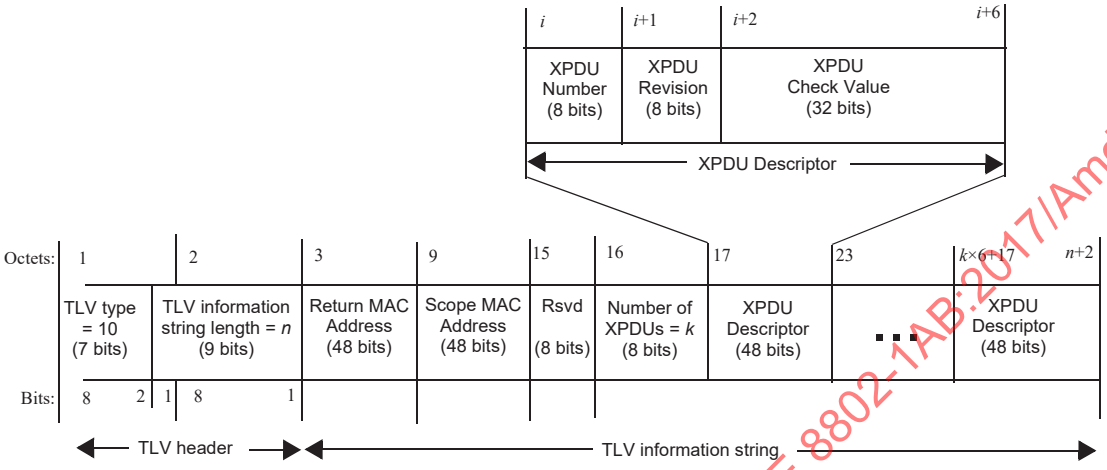

Figure 8-11b—Extension Request TLV format

8.5a.2.1 TLV information string length

The TLV information string length field shall contain the number of octets in the TLV information string.

8.5a.2.2 Return MAC Address

The individual MAC address of the MSAP supporting the LLDP agent, to be used as the destination address for the requested Extension LLDPDUs.

8.5a.2.3 Scope MAC Address

The MAC Address used by this LLDP agent as the destination address for Normal LLDPDUs.

8.5a.2.4 Number of XPDUs

The number of XPDUs field contains the count of the XPDU Descriptors in this TLV. Since the maximum size of the TLV information string is 511 octets, the minimum number of XPDU Descriptors is 1 and the maximum number is 82. When the number of XPDU Descriptors is k, the TLV information string length is at least $k \times 6 + 14$, but it need not be exactly $k \times 6 + 14$. This allows an implementation to use a fixed length TLV information string with a variable number of XPDU Descriptors.

8.5a.2.5 XPDU Descriptor

An XPDU Descriptor comprises an 8-bit XPDU Number, an 8-bit XPDU Revision, and a 32-bit XPDU Check, as specified in 8.5a.1.5.

8.5a.2.6 Extension Request TLV usage rules

An Extension Request LLDPDU shall contain an Extension Request TLV as the third TLV, and shall not contain more than one Extension Request TLV. An Extension Request TLV shall not be contained in a

Normal LLDPDU or an Extension LLDPDU. Each XPDU Descriptor associated with a MSAP identifier and Scope MAC Address shall have a unique XPDU Number.

8.5a.3 Extension Identifier TLV

The Extension Identifier TLV identifies an XPDU containing one or more TLVs that would not fit in the Normal LLDPDU transmitted by this LLDP agent. The format for this TLV is shown in Figure 8-11c.

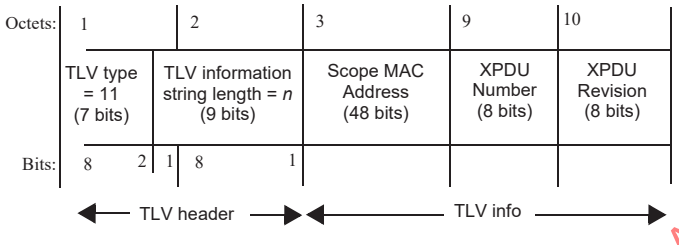

Figure 8-11c—Extension Identifier TLV format

8.5a.3.1 TLV information string length

The TLV information string length field shall contain the number of octets in the TLV information string.

8.5a.3.2 Scope MAC Address

The MAC Address used by this LLDP agent as the destination address for Normal LLDPDUs.

8.5a.3.3 XPDU Number

The XPDU Number portion of the XPDU Descriptor specified in 8.5a.1.5.

8.5a.3.4 XPDU Revision

The XPDU Revision portion of the XPDU Descriptor specified in 8.5a.1.5.

8.5a.3.5 Extension Identifier TLV usage rules

An Extension LLDPDU shall contain an Extension Identifier TLV as the third TLV, and shall not contain more than one Extension Identifier TLV. An Extension Identifier TLV shall not be contained in a Normal LLDPDU or an Extension Request LLDPDU.

8.6 Organizationally Specific TLVs

Change the heading and content of 8.6.1 as follows:

8.6.1 ~~Basic~~ Organizationally Specific TLV format

The ~~basic~~ format for Organizationally Specific TLVs is shown in Figure 8-12.

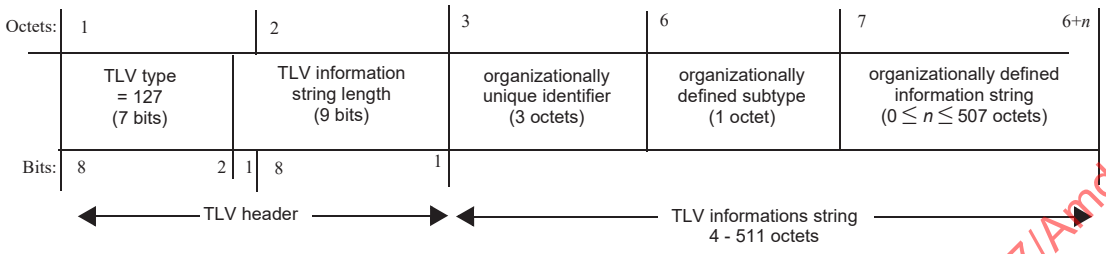

Figure 8-12—~~Basic format~~ Format for Organizationally Specific TLVs

The following subclauses indicate how the individual fields shall be defined.

9. LLDP agent operation

Change the introduction to Clause 9 as follows:

This clause describes the operation of the protocol from the point of view of a single LLDP agent (see Clause 6); i.e., it describes the operation of the protocol for the transmission and reception of LLDPDUs on a single Port~~, using a MAC address (see Clause 7) as the destination address~~. Where the use of more than one Scope MAC Address (7.1.1) ~~address~~ is supported on a given Port, a separate instance of the LLDP agent is responsible for the operation of the protocol for each Scope MAC Address ~~address~~ that is supported.

9.1 Overview

Change 9.1 including Table 9-1 as follows:

Each LLDP agent is responsible for causing the following tasks to be performed:

a) Maintaining current information in the LLDP local system MIB.

b) Extracting and formatting LLDP local system MIB information for transmission to the remote port LLDP agent(s) at regular intervals or whenever there is a change in the system condition or status.

c) Recognizing and processing received LLDP frames.

d) Maintaining current values in the LLDP remote system MIB.

e) Using the procedure somethingChangedLocal() and the procedure somethingChangedRemote() to notify the PTOPO MIB manager and MIB managers of other optional MIBs whenever a value or status change has occurred in one or more objects in the LLDP local system LLDP remote systems MIB.

NOTE 1—A consequence of the support of multiple MAC addresses on a Port is that there are multiple copies of the LLDP local system MIB and remote system MIB, one copy for each address supported.

LLDP allows implementations that support three different operating modes: transmit only, receive only, or both transmit and receive. An LLDP agent shall conform to the specifications of each of the state machines indicated in Table 9-1 for the operating mode that it supports.

Table 9-1—Subclause/operating mode applicability

Subclause number	Subclause title	Transmit only	Receive only	Transmit and receive
9.2.9	Receive state machine	Mandatory if XLLDP is supported	Mandatory	Mandatory
9.2.8	Transmit state machine	Mandatory	—	Mandatory
9.2.10	Transmit timer state machine	Mandatory	—	Mandatory

NOTE 2—When XLLDP is supported and the operating mode is transmit only, the receive state machine is only enabled to recognize and respond to Extension Request LLDPDUs. Information advertised by remote LLDP agents is not stored in the LLDP remote systems MIB.

9.1.1 Frame transmission

Change 9.1.1 and 9.1.2 as follows (this results in the removal of the heading for 9.1.2 and the renumbering of subsequent headings):

An LLDPDU transmission cycle consists of the transmission of a Normal LLDPDU, potentially including a Manifest TLV if the implementation supports XLLDP, and the subsequent transmission of one or more Extension LLDPDUs upon receipt of Extension Request LLDPDUs. For an implementation not supporting XLLDP, the entire transmission cycle consists of the transmission of a single Normal LLDPDU.

The LLDP local system MIB contains the information needed for constructing individual TLVs and includes a capability that allows network managers to select the optional TLVs to be included in the Normal LLDPDU and, if XLLDP is supported, in one or more Extension LLDPDUs (see 10.2.2).

9.1.1.1 Normal LLDPDUs

The transmission of Normal LLDPDUs ~~LLDPDU transmission~~ is performed by the transmit state machine, and the timing of transmission cycles ~~transmissions~~ is controlled by the transmit timer state machine. Transmissions occur as a result of a number of different local events, as follows:

a) A regular background transmission cycle occurs as a result of a transmission timer expiring. This prevents the receiving system from timing ~~ensures that the receiving system does not time~~ out the information received, as a result of the "time to live" associated with the last received LLDPDU expiring.

b) If a *new neighbor* is recognized (i.e., an LLDPDU is received by the receive state machine from a hitherto unknown remote LLDP agent), then four LLDPDU transmission cycles ~~transmissions~~ are performed at shorter time intervals to quickly update the new neighbor ~~ensure that the new neighbor is quickly updated~~ with current information about its own neighbors. This rapid transmission behavior is suppressed if the remote systems MIB is not able to accommodate the new neighbor information without removing current information relating to an older neighbor, in order to prevent excessive PDU transmissions resulting from "churn" effects.

c) If a condition (status or value) change occurs in one or more objects in the LLDP local system MIB, then a transmission cycle is initiated ~~an LLDPDU is transmitted~~ immediately, in order to communicate the local changes as quickly as possible to any neighboring systems. The transmit timer state machine operates a simple credit-based transmission strategy that allows a number of transmission cycles ~~LLDPDUs to be transmitted~~ in quick succession (the maximum number being determined by the maximum credit value), thus allowing rapid updating of information in situations where the local state is changing quickly. However, once the LLDP ~~Agent's~~ agent's transmission credit is exhausted, the rate of transmission is cut to a maximum of one transmission cycle per second.

d) The overall timing of regular transmission cycles ~~transmissions~~ is governed by a system "tick" that operates on a nominal timebase of 1 s. However, in shared media LANs, in order to avoid bunching of transmissions, the intervals between ticks include a random jitter component so that, while the average time interval between ticks is 1 s, the interval between adjacent pairs of ticks can vary (see 9.2.2).

In order to reduce ~~minimize~~ synchronization effects, it is recommended that transmission cycle intervals for different LLDP agents on the same multi-port implementation are staggered.

~~**9.1.2 LLDPDU types**~~

~~The following two types of LLDPDUs are defined:~~

~~a) Normal LLDPDUs that provide management information about the local station to that station's neighbors.~~
~~b) A special shutdown advisory LLDPDU indicating that any information about the local station that is maintained in a remote LLDP agent's remote systems MIB is now invalid and is to be discarded.~~

~~9.1.2.1 Normal LLDPDUs~~

~~The LLDP local system MIB contains the information needed for constructing individual TLVs and includes a capability that allows network managers to select the optional TLVs to be included in the LLDPDU (see 10.2.2).~~

When the transmit state machine (9.2.8) initiates a transmission cycle, ~~determines that a normal LLDPDU is to be transmitted,~~ the LLDP MIB manager extracts the selected information from the LLDP local system MIB and constructs ~~an~~ a Normal LLDPDU ~~as defined in 9.1,~~ containing the following:

a) The mandatory TLVs as specified in 8.2:
   1) Chassis ID TLV (see 8.5.2).
   2) Port ID TLV (see 8.5.3).
   3) Time To Live TLV, with the TTL value set equal to txTTL (see 8.5.4).
b) A Manifest TLV (see 8.5a.1), if XLLDP is supported and any of the selected information is to be included in an Extension LLDPDU. The Manifest TLV includes a descriptor for each Extension LLDPDU, even if the information selected to be included in that Extension LLDPDU has not changed.
c) ~~b)~~ Additional optional TLVs, from the basic management set or from one or more organizationally specific sets, as allowed by LLDPDU length restrictions and as selected in the LLDP local system MIB by network management (see Table 8-1).
d) ~~c)~~ Optionally, an End Of LLDPDU TLV.

Optional TLVs from the basic management set, such as the Management Address TLV, with different TLV type and subtype combinations as well as optional TLVs with different ~~OUI~~ organizationally unique identifier and organizationally defined subtype combinations from one or more organizationally specific sets can be included within the same LLDPDU.

9.1.1.2 ~~9.1.2.2~~ Shutdown LLDPDUs

A special procedure exists for the case in which a LLDP agent knows an associated port is about to become non-operational (for example, the adminStatus for the port is transitioning to 'disabled'). In the event a port, currently configured with LLDP frame transmission enabled, either becomes disabled for LLDP activity, or the interface is administratively disabled, the transmit state machine attempts to send a final LLDP shutdown LLDPDU with:

a) The Chassis ID TLV.
b) The Port ID TLV.
c) The Time To Live TLV with the TTL field set to zero.
d) Optionally, an End Of LLDPDU TLV.

The shutdown LLDPDU does not include any other optional TLVs and, if possible, should be transmitted before the interface is disabled.

NOTE—There is an inherent race condition between an interface knowing it is going down and its ability to send "one more frame." If possible, the actual transition of the port to disabled is delayed until after this frame is transmitted. In the event where adminStatus is transitioning to the disabled state and the LLDP agent is shutting down, this shutdown procedure should be executed for all local ports.

9.1.1.3 Extension Request LLDPDUs

Extension Request LLDPDUs are transmitted in response to the reception of a Manifest LLDPDU. The transmission is controlled by the extended receive state machine. When XLLDP is supported, and a Manifest LLDPDU or an Extension LLDPDU is received, and the Manifest TLV includes XPDU Descriptors that do not match the XPDU Descriptors in the LLDP remote systems MIB entry for the MSAP Identifier received in the LLDPDU, the LLDP MIB manager constructs an Extension Request LLDPDU containing the following:

a) The mandatory TLVs as specified in 8.2:
   1) Chassis ID TLV (8.5.2).
   2) Port ID TLV (8.5.3).
   3) Extension Request TLV with a list of one or more XPDU descriptors (8.5a.2).
b) Optionally, an End Of LLDPDU TLV.

9.1.1.4 Extension LLDPDUs

Extension LLDPDUs are transmitted in response to the reception of an Extension Request LLDPDU. The transmission is controlled by the receive state machine. When XLLDP is supported, and an Extension Request LLDPDU is received, then for each XPDU Number and XPDU Revision pair in the Extension Request TLV that matches an XPDU Number and XPDU Revision in the LLDP local system MIB, the LLDP MIB manager extracts the selected information from the LLDP local system MIB and constructs an Extension LLDPDU containing the following:

a) The mandatory TLVs as specified in 8.2:
   1) Chassis ID TLV (8.5.2).
   2) Port ID TLV (8.5.3).
   3) Extension Identifier TLV with the requested XPDU Number and XPDU Revision pair (8.5a.3).
b) Additional optional TLVs, from the basic management set or from one or more organizationally specific sets, as allowed by LLDPDU length restrictions and as selected in the LLDP local system MIB by network management (Table 8-1).
c) Optionally, an End Of LLDPDU TLV.

9.1.2 Frame reception

Change 9.1.2 (re-numbered from 9.1.3 by the removal of the heading for 9.1.2 above) as follows:

LLDP frame reception consists of four ~~three~~ phases: frame recognition, frame validation, TLV collection, and LLDP remote systems MIB updating. ~~All error checking is done during frame validation.~~

9.1.2.1 Frame recognition

Frame recognition is performed at the LLDP/LSAP (see Figure 6-2), where the destination address and EtherType values determine whether the frame is an LLDP frame destined for this LLDP agent or not.

9.1.2.2 Frame validation

Frames that are recognized as LLDP frames are then validated to determine whether they are properly constructed and contain the correct set of mandatory TLVs. Frames that do not ~~; frames that~~ pass the validation criteria are discarded.

9.1.2.3 TLV collection

All received TLVs are validated by checking for format errors, and invalid TLVs are ignored. If the TLVs received in the Normal LLDPDU did not include a Manifest TLV (or the implementation does not support XLLDP and does not recognize the Manifest TLV), then all validated TLVs are passed to the LLDP remote systems MIB immediately. If the implementation supports XLLDP and the Normal LLDPDU includes a Manifest TLV, then this TLV is used to determine which Extension LLDPDUs need to be received before a complete set of TLVs can be passed to the LLDP remote systems MIB.

An implementation supporting XLLDP compares the XPDU descriptors in the Manifest TLV to those received in a previous Manifest TLV to determine which XPDUs (identified by the XPDU Number) contain new information. If no Manifest TLV has been received previously, then all of the XPDUs described in the Manifest TLV contain new information. If there are no XPDUs that have new information, then the validated TLVs received in the Normal LLDPDU are passed to the LLDP remote systems MIB. Otherwise these TLVs are retained until the XPDUs with new information have been received.

If there are XPDUs that have new information, then an Extension Request LLDPDU requesting one or more of the XPDUs with new information, is transmitted to the LLDP agent that sourced the Normal LLDPDU. A timer is started when an Extension Request LLDPDU is transmitted, and the request is retried once if the timer expires before all requested XPDUs have been received. When all requested XPDUs have been received, another Extension Request LLDPDU is transmitted requesting one or more of the remaining XPDUs with new information. This process repeats until all XPDUs with new information have been received. If a new Manifest LLDPDU is received before all XPDUs have been received, any received XPDUs that have the same Revision and Checksum in the new Manifest TLV are retained, and any received XPDUs that have different Revision and Checksum are discarded. The extended receive state machine then continues to issue Extension Requests for any necessary XPDUs.

Received frames containing Extension LLDPDUs are recognized and validated following the same steps as frames containing Normal LLDPDUs. Once validated, the XPDU check value is calculated. This check value, together with the XPDU Number and XPDU Revision from the Extension Identifier TLV, is compared to the XPDU descriptors in the Manifest TLV. If there is a matching descriptor then all valid TLVs received in this Extension LLDPDU are retained. When all XPDUs with new information have been received, all of the retained TLVs are passed to the LLDP remote systems MIB.

9.1.2.4 LLDP remote systems MIB updating

The collected TLVs are used to create, modify, or delete an entry ~~update the contents of entries~~ in the LLDP remote systems MIB. The relevant entry in the LLDP remote systems MIB is determined by ~~for~~ the originating MSAP identifier formed from the contents of the Chassis ID and Port ID TLVs.~~LSAP identifier and destination MAC address.~~ The LLDP remote systems MIB is updated as follows:

a) If the TTL value in the Time To Live TLV is greater than zero ~~frame is a normal LLDPDU~~ and an entry does not exist in the LLDP remote systems MIB for the originating MSAP ~~LSAP~~ identifier and there is sufficient space (or sufficient space can be created by the steps in 9.2) for the new information ~~destination MAC address~~, then a new entry in the MIB is created~~, assuming that sufficient space exists to do so~~.

b) If the TTL value in the Time To Live TLV is greater than zero ~~frame is a normal LLDPDU~~ and an entry already exists in the remote systems MIB for the originating MSAP ~~LSAP~~ identifier and there is sufficient space (or sufficient space can be created by the steps in 9.2) for the new information ~~destination MAC address~~, then that entry is modified as follows:

   1) The TLVs received in the Normal LLDPDU are used to replace any information elements in the existing entry in the MIB that were previously received in a Normal LLDPDU. Note that if the Normal LLDPDU did not contain a Manifest TLV (or the implementation does not support

XLLDP and does not recognize the Manifest TLV), then the new information contained in the LLDPDU is used to replace the whole of the existing entry in the MIB; i.e., if there are information elements in the existing entry for which there are corresponding elements in the received LLDPDU, then those elements are updated using the information received; any other information elements in the existing entry in the MIB are deleted.

2) The TLVs received in an Extension LLDPDU are used to replace any information elements in the existing entry in the MIB that were previously received in an Extension LLDPDU with the same XPDU Number.

3) Any information elements in the existing entry in the MIB that were previously received in an Extension LLDPDU with an XPDU Number that is not included in the received Manifest TLV are deleted.

c) If the TTL value in the Time To Live TLV is greater than zero and an entry exists in the LLDP remote systems MIB for the originating MSAP identifier and there is insufficient space for the new information, then the entry is deleted from the MIB.

d) If the TTL value in the Time To Live TLV is equal to zero and an entry exists in the LLDP remote systems MIB for the originating MSAP identifier, then the entry is deleted from the MIB.

When an entry in the LLDP remote systems MIB is created or modified (even if the modification is to replace all information elements with the same information), the rxInfoTTL timer associated with that entry is set to the TTL value from the Time To Live TLV. ~~One of the parameters received in an incoming LLDPDU is the TTL value; this~~ This determines how long the information associated with that MSAP ~~LSAP~~ identifier and destination MAC address is to be stored in the LLDP remote systems MIB before being aged out and deleted. The ageing out of old data purges the MIB ~~ensures that the MIB is purged~~ of data that was originated by systems that are no longer neighbors as a result of topology changes, system failure, or system inactivation.

~~If the received frame is a shutdown LLDPDU and an entry exists in the LLDP remote systems MIB for the originating LSAP identifier and destination MAC address, then that entry is deleted from the MIB.~~

9.2 State machines

9.2.1 Notational conventions used in state diagrams

Change Table 9-2 as follows:

Table 9-2—State machine symbols

Symbol	Interpretation
( )	Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes.
;	Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal English language conventions for reading text.
=	Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., a = b = X the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator.
!	Logical NOT operator.

Table 9-2—State machine symbols *(continued)*

Symbol	Interpretation
&&	Logical AND operator.
||	Logical OR operator.
if...then...	Conditional action. If the Boolean expression following the **if** evaluates to TRUE, then the action following the **then** is executed.
{statement 1, ... statement N}	Compound statement. Braces are used to group statements that are executed together as if they were a single statement.
!=	Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.
==	Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right.
<	Less than. Evaluates to TRUE if the value of the expression to the left of the operator is less than the value of the expression to the right.
<=	Less than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either less than or equal to the value of the expression to the right.
>	Greater than. Evaluates to TRUE if the value of the expression to the left of the operator is greater than the value of the expression to the right.
>=	Greater than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either greater than or equal to the value of the expression to the right.
+	Arithmetic addition operator.
–	Arithmetic subtraction operator.

9.2.2 Timers

Insert new subclause 9.2.2.5 after 9.2.2.4 as follows:

9.2.2.5 rxTimer

An instance of this timer exists for each extended receive state machine to impose a time limit on each extension request, and on the overall TLV collection process.

9.2.5 Per-Agent variables

Delete 9.2.5.2, renumbering subsequent subclauses as necessary.

Change the title and text of 9.2.5.3 (re-numbered from 9.2.5.4 by the deletion of 9.2.5.2 above) as follows:

9.2.5.3 Scope MAC Address ~~address~~

The MAC address associated with this instance of the LLDP agent ~~Agent, i.e.,~~ that defines the scope of propagation of Normal LLDPDUs in the network. It is used as the destination MAC address ~~that is used~~ when the LLDP agent ~~Agent~~ transmits Normal LLDPDUs, and the destination MAC address that, when present in a received LLDPDU, causes the LLDPDU ~~PDU~~ to be passed to this LLDP agent instance for processing.

9.2.5.8 rcvFrame

Change 9.2.5.8 (re-numbered from 9.2.5.9 by the deletion of 9.2.5.2 above) as follows:

This variable indicates that an LLDP frame has been recognized by the LLDP LSAP function and is ready to be processed by the LLDP receive state machine. If both of the following conditions are TRUE, the variable rcvFrame is set to TRUE and the frame is made available to the receive state machine for validation and processing:

a) The destination address value is the Scope MAC Address ~~address~~ associated with this specific LLDP ~~Agent~~ agent or the individual MAC address of the MSAP supporting the LLDP agent.
b) The EtherType value carried by the frame is the LLDP EtherType defined in Table 7-3.

NOTE—The model that is assumed for reception is that incoming LLDPDUs are presented to all LLDP agents associated with the reception Port, and if the ~~destination~~ Scope MAC Address ~~address~~ is not one that is associated with a given LLDP agent ~~Agent~~, that LLDPDU is ignored by that LLDP agent. In practice, at most one LLDP agent processes any received LLDPDU, as there is only one LLDP agent associated with any given ~~destination~~ Scope MAC Address ~~address~~ on a given reception Port.

Insert new subclauses 9.2.5.14 and 9.2.5.15 after 9.2.5.13 (renumbered from 9.2.5.14), renumbering subsequent subclauses as necessary:

9.2.5.14 rxType

This variable indicates the type of the most recently received LLDP frame. It is set by the rxProcessFrame() procedure to one of the following values:

a) xreq: The LLDP agent supports XLLDP and identifies the frame as containing a request for extended PDUs.
b) xpdu: The LLDP agent supports XLLDP and identifies the frame as containing an extended PDU.
c) shutdown: The frame contains a shutdown LLDPDU (which has an rxTTL value equal to zero).
d) manifest: The LLDP agent supports XLLDP and the frame is a Normal LLDPDU that has a rxTTL value greater than zero and contains a Manifest TLV.
e) normal: The frame contains a Normal LLDPDU that has an rxTTL value greater than zero, and either the LLDP agent does not support XLLDP or the frame does not contain a Manifest TLV.
f) invalid: The frame is improperly formatted for a valid LLDPDU, or the frame contains an extended PDU or a request for extended PDUs but the LLDP agent does not support XLLDP.

9.2.5.15 sentManifest

This Boolean variable is set to TRUE when the transmit state machine generates a Manifest LLDPDU, and is set to FALSE when the transmit state machine generates a Normal LLDPDU that does not contain a Manifest TLV. It is used to enable the receive state machine to recognize Extension Request LLDPDUs and respond with Extension LLDPDUs.

Insert new subclause 9.2.5a after 9.2.5 and all of its subclauses as follows:

9.2.5a Per-Neighbor variables

An instance of each of these variables exists per neighbor (identified by an MSAP) for each LLDP agent; they are available to all of the state machines associated with an LLDP agent.

9.2.5a.1 createExtRx

This Boolean is used to initialize a per-neighbor extended receive state machine. It is set to TRUE when an extended receive state machine is created in response to the receipt of a Manifest LLDPDU from a neighbor for which an extended receive state machine does not already exist. It is set to FALSE by the extended receive state machine following initialization.

9.2.5a.2 rxExtended

This Boolean is used to pass control between the per-agent receive state machine and the per-neighbor extended receive state machine. It is set to TRUE by the receive state machine when an LLDPDU containing a Manifest TLV or an Extension Identifier TLV is received, and is set to FALSE by the extended receive state machine.

9.2.5a.3 manifestComplete

This Boolean indicates to the receive state machine whether the extended receive state machine has received all TLVs necessary to update the information for this MSAP in the LLDP remote systems MIB.

9.2.5a.4 xreqComplete

This Boolean is used within the extended receive state machine to indicate that all XPDUs requested for this MSAP have been received.

9.2.5a.5 rxTick

This Boolean is set TRUE by the system timer tick at one second intervals (see 9.2.2). It is set FALSE by the operation of the extended receive state machine.

9.2.5a.6 rxTTLExpired

This Boolean is used to indicate that the extended receive state machine has been unable to collect all the TLVs within a time interval greater than the received rxTTL value. It is set to TRUE when the rxTimer is decremented to zero and either all Extension LLDPDUs for the most recently transmitted Extension Request LLDPDU have been received, or that Extension Request LLDPDU was a retry of a previous request.

9.2.7 State machine procedures

9.2.7.4 mibDeleteObjects()

Change 9.2.7.4 as follows:

The mibDeleteObjects() procedure deletes all information in the LLDP remote systems MIB associated with a given MSAP identifier (and destroys the associated extended receive state machine if one exists) if an LLDPDU is received with an rxTTL value of zero (see 9.2.7.7.1), the extended receive state machine sets rxTTLExpired to TRUE, or the timing counter rxInfoTTL expires (see 9.2.2.1).

9.2.7.5 mibUpdateObjects()

Change 9.2.7.5 as follows:

The mibUpdateObjects() procedure updates the MIB objects corresponding to the TLVs contained in the received Normal LLDPDU, and all TLVs contained in any received Extension LLDPDUs, for the LLDP remote system if MIB space is available, as follows:

a) Compare the MSAP identifier in the current LLDPDU with the MSAP identifiers in the LLDP remote systems MIB:

1) If a match is found, but no difference exists between the information in the TLVs just received and the information in the LLDP remote systems MIB, then set the control variable rxChanges to FALSE, and set the timing counter rxInfoTTL associated with the MSAP identifier to rxTTL.

2) If a match is found and sufficient space is not available in the LLDP remote systems MIB for the updated information in the TLVs just received, then follow the tooManyNeighbors procedure (9.2.7.5.1) to determine whether to discard the TLVs from a new neighbor or to delete information from an existing neighbor that is already in the LLDP remote systems MIB,

3) ~~1)~~ If a match is found~~, replace~~ and sufficient space is available (or has been made available) in the LLDP remote systems MIB for the updated information in the TLVs just received, then:

i) Replace all ~~current~~ information associated with the MSAP identifier in the LLDP remote systems MIB that was previously received in a Normal LLDPDU with the information in the ~~current~~ just received Normal LLDPDU.

ii) For each received Extension LLDPDU associated with the MSAP identifier, if any, replace all information in the LLDP remote systems MIB that was previously received in an Extension LLDPDU of the same XPDU Number with the information in the just received Extension LLDPDU. If a received Extension LLDPDU has an XPDU Number that has not been previously received, add the new information to the LLDP remote system MIB entry.

iii) Set the control variable rxChanges to TRUE.

iv) Set the timing counter rxInfoTTL associated with the MSAP identifier to rxTTL.

4) If no match is found and sufficient space is not available in the LLDP remote systems MIB, then follow the tooManyNeighbors procedure (9.2.7.5.1) to determine whether to discard the TLVs from a new neighbor or to delete information from an existing neighbor that is already in the LLDP remote systems MIB,

5) ~~2)~~ If no match is found and sufficient space is available (or has been made available), create a new MIB structure to receive information associated with the new MSAP identifier, ~~and~~ set these MIB objects to the values indicated in their respective TLVs, set the control variable rxChanges to TRUE, and set the timing counter rxInfoTTL associated with the MSAP identifier to rxTTL.

~~b) Set the timing counter rxInfoTTL associated with the MSAP identifier to rxTTL.~~

If an incoming TLV is not recognized by the receiving LLDP agent, the TLV is stored in the LLDP remote systems MIB as follows:

b) ~~c)~~ If the TLV type value is in the range of the reserved TLV types in Table 8-1, the TLV can be from a later version of the basic management set and is stored according to the basic TLV format shown in Figure 8-2. These TLVs are indexed by their TLV type.

c) ~~d)~~ If the TLV type value is 127, the TLV is an Organizationally Specific TLV and is stored according to the basic format for Organizationally Specific TLVs shown in Figure 8-12. These TLVs are indexed by their ~~OUI~~ organizationally unique identifier and organizationally defined TLV subtype.

Insert new subclause 9.2.7.5.1 at the end of 9.2.7.5 as follows:

9.2.7.5.1 Too many neighbors

When there is insufficient space in the LLDP remote systems MIB to store information from a new neighbor something needs to be discarded. Either the received LLDPDU is discarded or existing information within the current remote systems MIB is discarded in order to make space for the new information received in the LLDPDU:

a) Set the flag variable tooManyNeighbors to TRUE.
b) If the information selected to be discarded is the information in the current LLDPDU:
   1) Set the tooManyNeighborsTimer as specified in Equation (3).

$$\text{tooManyNeighborsTimer} = \max(\text{tooManyNeighborsTimer}, \text{rxTTL}) \quad (3)$$

   where rxTTL is the TTL value in the current LLDPDU
   2) Discard the current LLDPDU and increment the statsFramesDiscardedTotal counter.
   3) Set the control variable rxChanges to FALSE. Wait for the next LLDPDU.
c) If the information selected to be discarded is currently in the LLDP remote systems MIB:
   1) Delete all information associated with the selected neighbor's MSAP identifier from the LLDP remote systems MIB, and destroy the associated extended receive state machine if one exists.
   2) Set the tooManyNeighborsTimer as specified in Equation (4).

$$\text{tooManyNeighborsTimer} = \max(\text{tooManyNeighborsTimer}, \text{rxInfoTTL}) \quad (4)$$

   where rxInfoTTL = the selected neighbor's TTL value
   3) If sufficient space exists to store the information received in the current LLDPDU in the LLDP remote systems MIB, set control variable rxChanges to TRUE.
   4) If sufficient space still does not exist to store the information received in the current LLDPDU in the LLDP remote systems MIB, perform steps 1–3 again.

The variable tooManyNeighbors is automatically set to FALSE whenever the tooManyNeighborsTimer expires.

NOTE—The tooManyNeighborsTimer is not precise, as it is only cleared (and the tooManyNeighbors variable set FALSE) by timer expiration, and not by the removal of the too many neighbors condition.

9.2.7.7 rxProcessFrame()

Change 9.2.7.7 as follows:

The rxProcessFrame() procedure:

a) Validates the LLDPDU and sets the value of rxType as specified in 9.2.7.7.1.
b) ~~a)~~ Validates the TLVs contained in the LLDPDU as defined in 9.2.7.7.1 through 9.2.8.7.3.
~~b) Determines whether or not a MIB update may be required as defined in 9.2.8.7.4.~~
~~c) If sufficient space is not available, determines whether to discard the incoming LLDPDU from a new neighbor or to delete information from an existing neighbor that is already in the LLDP remote systems MIB, as defined in 9.2.7.7.5. The tooManyNeighborsTimer and the tooManyNeighbors flag variable are both set during this process.~~

~~NOTE—The flag variable tooManyNeighbors is automatically reset when the tooManyNeighborsTimer expires.~~

9.2.7.7.1 LLDPDU validation

Change 9.2.7.7.1 as follows:

The receive module processes each incoming LLDPDU as it is received. The statsFramesInTotal counter for the port is incremented and the LLDPDU is checked to verify the presence of the three mandatory TLVs at the beginning of the LLDPDU as defined in 8.2.

a) The first TLV is extracted:
   1) If the extracted TLV type value does not ~~equal 1, the TLV is not a~~ match the Chassis ID TLV type:

i) The LLDPDU is discarded.

ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

iii) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~

iv) The procedure rxProcessFrame() is terminated.

2) If the extracted TLV type value ~~equals 1~~ matches the Chassis ID TLV type, and the chassis ID TLV information string length is not within the range $2 \leq n \leq 256$:

i) The LLDPDU is discarded.

ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

iii) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~

iv) The procedure rxProcessFrame() is terminated.

3) If the extracted TLV type value ~~equals 1~~ matches the Chassis ID TLV type, and the chassis ID TLV information string length is within the range $2 \leq n \leq 256$, the chassis ID value is extracted to become the first part of the MSAP identifier.

b) The second TLV is extracted:

1) If the extracted TLV type value does not ~~equal 2, the TLV is not a~~match the Port ID TLV type:

i) The LLDPDU is discarded.

ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

iii) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~

iv) The procedure rxProcessFrame() is terminated.

2) If the extracted TLV type value ~~equals 2~~ matches the Port ID TLV type, and the port ID TLV information string length is not within the range $2 \leq n \leq 256$:

i) The LLDPDU is discarded.

ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

iii) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~

iv) The procedure rxProcessFrame() is terminated.

3) If the extracted TLV type value ~~equals 2~~ matches the Port ID TLV type, and the port ID TLV information string length is within the range $2 \leq n \leq 256$, the port ID value is extracted and appended to the chassis ID value to complete construction of the MSAP identifier.

c) The third TLV is extracted:

1) If the extracted TLV type value does not match the Time To Live, Extension Request, or Extension TLV type ~~equal 3~~, the LLDPDU is invalid: ~~TLV is not a Time To Live TLV.~~

i) The LLDPDU is discarded.

ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

iii) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~

iv) The procedure rxProcessFrame() is terminated.

2) If the extracted TLV type value matches the Time To Live or Extension TLV type, and the adminStatus variable is enableTxOnly, the LLDPDU is discarded because the receive state machine is only enabled to receive Extension Request LLDPDUs.

i) The LLDPDU is discarded.

ii) The rxType value is set equal to invalid.

iii) The procedure rxProcessFrame() is terminated.

3) ~~2)~~ If the extracted TLV type value matches the Time To Live TLV type ~~equals 3~~, and the Time To Live TLV information string length is less than 2:

   i) The LLDPDU is discarded.

   ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

   iii) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~

   iv) The procedure rxProcessFrame() is terminated.

4) ~~3)~~ If the extracted TLV type value matches the Time To Live TLV type ~~equals 3~~, and the Time To Live TLV information string length is greater than or equal to 2; ~~the~~

   i) The first two octets of the TLV information string is extracted and rxTTL is set to this value. ~~:~~

   ii) ~~i)~~ If rxTTL equals zero, the rxType value is set equal to shutdown and the procedure rxProcessFrame() is terminated. ~~a shutdown frame has been received. The MSAP identifier and rxTTL are passed up to the LLDP MIB manager, and further LLDPDU validation is terminated.~~

   iii) ~~ii)~~ If rxTTL is non-zero, the rxType value is set to normal. ~~LLDPDU validation continues and the remaining TLVs are validated.~~

5) If the extracted TLV type value matches the Extension Request TLV type, XLLDP is supported, and the received MSAP identifier and Scope MAC Address match the MSAP and Scope MAC Address of this LLDP agent:

   i) The rxType value is set equal to xreq.

   ii) The procedure rxProcessFrame() is terminated.

6) If the extracted TLV type value matches the Extension Request TLV type, and either XLLDP is not supported or the received MSAP identifier and Scope MAC Address combination does not match the MSAP and Scope MAC Address combination of this LLDP agent:

   i) The LLDPDU is discarded.

   ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

   iii) The rxType value is set equal to invalid.

   iv) The procedure rxProcessFrame() is terminated.

7) If the extracted TLV type value matches the Extension TLV type, XLLDP is supported, and an extended receive state machine exists for the neighbor identified by the received MSAP and Scope MAC Address:

   i) The rxType value is set equal to xpdu.

8) If the extracted TLV type value matches the Extension TLV type, and either XLLDP is not supported or an extended receive state machine does not exist for the neighbor identified by the received MSAP and Scope MAC Address:

   i) The LLDPDU is discarded.

   ii) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

   iii) The rxType value is set equal to invalid.

   iv) The procedure rxProcessFrame() is terminated.

Each of the remaining TLV information elements are decoded in succession as required by their particular TLV format definitions:

d) If the TLV type value equals 0, the TLV is the End Of LLDPDU TLV, and the procedure rxProcessFrame() is terminated. ~~The MSAP identifier, rxTTL, and all validated TLVs are passed to the LLDP management entity for LLDP remote systems MIB updating.~~

e) If the TLV_type_value is less than 127 and is recognized by the LLDP implementation, it ~~(0 < TLV_type_value ≤ 8) the TLV is a member of the basic management set and~~ is validated according to the general rules for all TLVs defined in 9.2.7.7.2 as well as any specific rules defined for the particular TLVs defined in 8.5.

f) If the TLV_type_value is less than 127 and is not recognized by the LLDP implementation, it ~~in the range of reserved TLV types in Table 8-1, the TLV is unrecognized and~~ may be a basic TLV from a later LLDP version. The statsTLVsUnrecognizedTotal counter is incremented, and the TLV is assumed to be validated.

g) If the implementation supports XLLDP and the TLV type value matches the Manifest TLV type:

1) If the MSAP identifier in the current LLDPDU does not match an MSAP identifier in the LLDP remote systems MIB, the total MIB entry size is extracted from the Manifest TLV. If there is insufficient space in the LLDP remote systems MIB to store the neighbor information, and the policy is to discard the current LLDPDU rather than delete information for other MSAPs:

i) The flag variable tooManyNeighbors is set to TRUE.

ii) If the rxTTL value in the LLDPDU is greater than the value of the tooManyNeighborsTimer, the tooManyNeighborsTimer is set to the rxTTL value.

iii) The statsFrameDiscardedTotal counter is incremented.

iv) The rxType value is changed to invalid.

2) If the MSAP identifier in the current LLDPDU matches an MSAP identifier in the LLDP remote systems MIB, or there is sufficient space in the LLDP remote systems MIB to store the new neighbor information, then:

i) The rxType value is changed to manifest.

ii) If an extended receive state machine does not already exist for the MSAP identifier in the current LLDPDU, an extended receive state machine for that MSAP identifier is created and its createExtRx variable is set to TRUE.

h) ~~g)~~ If the TLV type value is 127, the TLV is an Organizationally Specific TLV:

1) If the TLV's ~~OUI~~ organizationally unique identifier and organizationally defined subtype are recognized, the TLV is validated according to the general rules for all TLVs defined in 9.2.7.7.2 as well as the general rules for Organizationally Specific TLVs defined in 9.2.8.7.3 plus any specific rules defined for the particular TLV.

2) If the TLV's ~~OUI~~ organizationally unique identifier and/or organizationally defined subtype are not recognized, the statsTLVsUnrecognizedTotal counter is incremented, and the TLV is assumed to be validated.

i) ~~h)~~ If the end of the LLDPDU has been reached the procedure rxProcessFrame() is terminated~~, the MSAP identifier, rxTTL, and all validated TLVs are passed to the LLDP management entity for LLDP remote systems MIB updating~~.

9.2.7.7.2 General validation rules for all TLVs

Change 9.2.7.7.2 as follows:

The value in the TLV information string length field is the value used to validate the TLV and to indicate the location of the next TLV in the LLDPDU.

All TLVs are subject to the general validation rules listed in this subclause as well as any specific usage rules defined for the particular TLV (for example, see systems capabilities TLV usage rules in 8.5.8.3):

a) If the LLDPDU contains more than one Chassis ID TLV, Port ID TLV, or Time To Live TLV:
   1) The LLDPDU is discarded.
   2) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.
   3) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~
   4) The procedure rxProcessFrame() is terminated.

b) If the implementation supports XLLDP and the LLDPDU contains more than one Manifest TLV, Extension Request TLV, or Extension Identifier TLV:
   1) The LLDPDU is discarded.
   2) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.
   3) The rxType value is set equal to invalid.
   4) The procedure rxProcessFrame() is terminated.

c) ~~b)~~ If the TLV information string length value is not greater than or equal to the sum of the lengths of all fields contained in the TLV information string:
   1) The LLDPDU is discarded.
   2) The statsFramesDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.
   3) The rxType value is set equal to invalid. ~~The variable badFrame is set to TRUE.~~
   4) The procedure rxProcessFrame() is terminated.

d) ~~c)~~ If any TLV contains an error condition specified for that particular TLV:
   1) The TLV is discarded.
   2) The statsTLVsDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.
   3) The location of the next TLV is based on the TLV information string length value of the current TLV.

e) ~~d)~~ With the exception of the TTL TLV for which specific rules apply (see 9.2.7.7.1), if the length of any field of a TLV is outside the length range specified for that particular TLV (for example, the TLV information string length is greater than the maximum permitted length for the TLV information string as specified for that TLV):
   1) The TLV is discarded.
   2) The statsTLVsDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.
   3) The location of the next TLV is based on the TLV information string length value of the current TLV.

f) ~~e)~~ If any TLV extends past the physical end of the frame:
   1) The TLV is discarded.
   2) The statsTLVsDiscardedTotal and statsFramesInErrorsTotal counters are both incremented.

g) ~~f)~~ Any information following an End Of LLDPDU TLV is ignored.

NOTE—Usage rules for individual TLVs allow some TLVs to appear more than once in an LLDPDU. Duplicate TLVs result in any one of the values being placed in the MIB, can cause the discard stats to increment, and can cause the change marker for the MIB entry to change if any of the TLV copies change the value even if the value finally recorded is unchanged. The only thing guaranteed is that the MIB value is set to one (unspecified) of the TLV values, and if that value is different to what was previously in the MIB then the change marker is set.

Delete 9.2.7.7.4 and 9.2.7.7.5.

9.2.7.11 txFrame()

Change 9.2.7.11 as follows:

The txFrame() procedure makes use of the services of LLC to transmit the LLDPDU constructed by either the mibConstrInfoLLDPDU(), ~~or~~ mibConstrShutdownLLDPDU(), constructXreqPDU(), or rxProcessXreq() procedures (9.2.8.2, 9.2.8.3, 9.2.7.16, 9.2.7.18). For Normal LLDPDUs, the ~~The~~ destination MAC address used is the MAC address associated with the instance of the LLDP agent ~~Agent~~ (see ~~7.1~~ 7.1.1 and 9.2.5.3). For Extension Request and Extension LLDPDUs, the destination MAC address used is the Return MAC Address in the received Manifest TLV or Extension Request TLV respectively (see 8.5a.1.2 and 8.5a.2.2). The source MAC address used is the individual MAC address of the transmitting port. The EtherType used is the LLDP EtherType identified in Table 7-3.