INTERNATIONAL STANDARD

ISO/ IEC/IEEE 8802-1CS

First edition
2022-07

Telecommunications and exchange between information technology systems — Requirements for local and metropolitan area networks —

Part 1CS: **Link-local registration protocol**

Télécommunications et échange entre systèmes informatiques — Exigences pour les réseaux locaux et métropolitains —

Partie 1CS: Protocole d'enregistrement de type liaison locale

Reference number
ISO/IEC/IEEE 8802-1CS:2022(E)

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 8802-1CS was prepared by the LAN/MAN of the IEEE Computer Society (as IEEE 802.1CS™-2020) 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 802.1CS™–2020

IEEE Standard for Local and Metropolitan Area Networks—

Link-local Registration Protocol

Developed by the

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 3 December 2020

IEEE SA Standards Board

Abstract: Protocols, procedures, and managed objects for a Link-local Registration Protocol (LRP) to replicate a registration database from one end to the other of a point-to-point link and to replicate changes to parts of that database are specified in this standard. A facility is provided to purge the replicated database if the source becomes unresponsive. LRP is optimized for databases on the order of 1 Mbyte.

Keywords: Bridged Local Area Networks, bridges, bridging, IEEE 802®, IEEE 802.1CS™, IEEE 802.1Q™, Link-local Registration Protocol, local area networks (LANs), LRP, MAC Bridges, Time-Sensitive Networking, TSN, Virtual Bridged Local Area Networks (virtual LANs)

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

PDF:	ISBN 978-1-5044-7253-1	STD24533
Print:	ISBN 978-1-5044-7254-8	STDPD24533

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. An IEEE Account is needed to access the application.

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

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.

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. 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. 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.

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at 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 standard was submitted to the IEEE SA for approval, the Higher Layer LAN Protocols Working Group had the following membership:

Introduction

This introduction is not part of IEEE Std 802.1CS-2020, IEEE Standard for Local and Metropolitan Area Networks—Link-local Registration Protocol.

This standard defines the Link-local Registration Protocol.

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

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

Contents

IEEE Standard for Local and Metropolitan Area Networks—

Link-local Registration Protocol

1. Overview

1.1 Scope

This standard specifies protocols, procedures, and managed objects for a Link-local Registration Protocol (LRP) to replicate a registration database from one end to the other of a point-to-point link and to replicate changes to parts of that database. A facility is provided to purge the replicated database if the source becomes unresponsive. Provision is made for a proxy system to operate LRP on behalf of a controlled system. LRP is optimized for databases on the order of 1 Mbyte.

1.2 Purpose

LRP is designed to facilitate the creation of applications that distribute information through all or part of a network.

1.3 State diagram conventions

This document uses the state diagram conventions defined in Annex E of IEEE Std 802.1Q-2018.[1] The programming language C (ISO/IEC 9899:2018) is also used to document the operation of conformant systems. C functions are distinguished with `this special fixed-width font` (e.g., 9.4.6).

1.4 Specification model

The model of operation documented by this standard is simply a basis for describing the functionality of a compliant equipment. Implementations can adopt any internal model of operation compatible with the externally visible behavior that this standard specifies. Conformance of equipment to this standard is purely in respect of observable protocol.

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

1.5 Note on inter-table references

A number of state machine variables, managed objects, and interface primitives in this standard use the phrase "a reference to *xyz*" (see e.g., 7.3.2.8). Such a reference might be implemented by a pointer, an integer index into an array, the value of some other object in *xyz* that is sufficiently unique to make the reference unambiguous. Other techniques are also possible. This is an implementation choice and is not specified by this standard. This use of "a reference to" is also used for managed objects in Clause 11. The actual data representations of references to managed objects in Clause 12 and Clause 13, of course, are not arbitrary.

1.6 Specification precedence

If any conflict among parts of this standard become apparent, state machine diagrams (see 1.3) take precedence over other parts of the standard, followed by information in normative tables, followed by that in normative text, followed by that in normative figures, followed by YANG and MIB modules. Non-normative tables, figures, and text are in annexes and are clearly marked as such.

1.7 Introduction

The Link-local Registration Protocol (LRP) provides an LRP Database Synchronization (LRP-DS) service interface (Clause 10) that can be used by one or more LRP applications in a network, each of which needs to distribute information to some or all of the relay systems and end systems in a network. The information is passed hop-by-hop, in the sense that each information element is associated with a specific pair of connected ports in the network.

LRP accomplishes its task by creating point-to-point bi-directional associations between systems' application instances, each association consisting of two one-way paths, and each path consisting of an applicant database at one end, and a registrar database at the other. LRP quickly and reliably replicates each applicant database to its neighbor's registrar database. Locally to a system, the application instance performs any propagation or modification of information among the applicant and registrar databases on different ports within a system. Globally, the application instances plus LRP can propagate information throughout a network. This standard does not specify any LRP application that might use LRP. These basic relationships are illustrated in Figure 1-1.

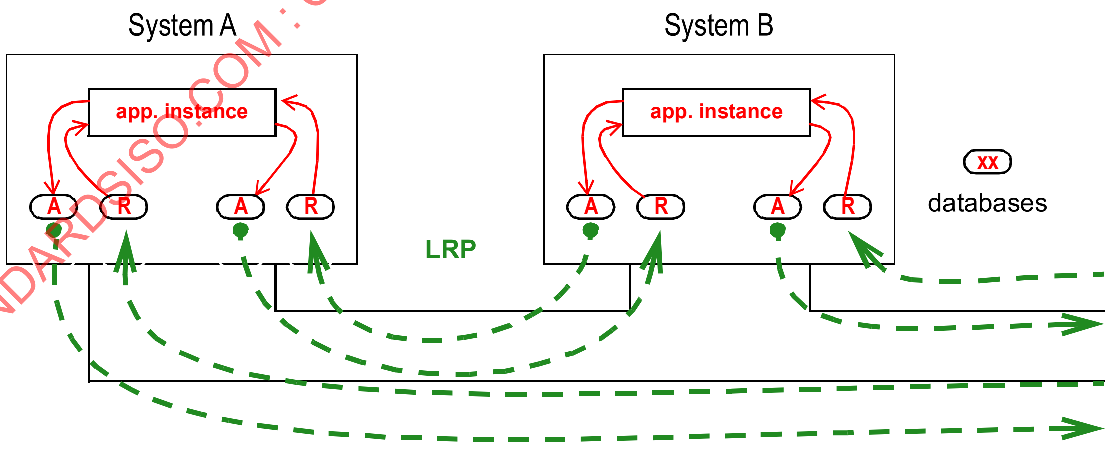


Figure 1-1—LRP simplified

LRP also supports a mode where a Proxy system operates the application instance, databases, and LRP on behalf of a Controlled system.

2. Normative references

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

Non-normative references (i.e., that provide additional information not required for the application of this document) are given in Annex D.

NOTE—The inclusion of a document in this list of normative references indicates that information in that document is necessary to implement the present standard. It does not imply that any other part of that referenced document is required to be implemented by a system conformant to the present standard.[2]

IEEE Std 802®, IEEE Standard for Local and metropolitan area networks—Overview and Architecture.[3, 4]

IEEE Std 802.1AB™-2016, IEEE Standard for Local and metropolitan area networks—Station and Media Access Control Connectivity Discovery.

IEEE Std 802.1Q™, IEEE Standard for Local and metropolitan area networks—Bridges and Bridged Networks.

IEEE Std 1003.1™, IEEE Standard for Information Technology—Portable Operating System Interface (POSIX™) Base Specifications, Issue 7.

IETF RFC 793, Transmission Control Protocol, September 1981.[5]

IETF RFC 2578, Std 58, Structure of Management Information Version 2 (SMIv2), April 1999.

IETF RFC 2579, Std 58, Textual Conventions for SMIv2, April 1999.

IETF RFC 2580, Std 58, Conformance Statements for SMIv2, April 1999.

IETF RFC 2863, The Interfaces Group MIB, June 2000.

IETF RFC 3232, Assigned Numbers: RFC 1700 is Replaced by an On-line Database, January 2002.

IETF RFC 4001, Textual Conventions for Internet Network Addresses, February 2005.

IETF RFC 6335, Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry, August 2011.

IETF RFC 7950, The YANG 1.1 Data Modeling Language, August 2016.

Internet Assigned Numbers Authority (IANA) database, Service Name and Transport Protocol Port Number Registry.[6]

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

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

[4] The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

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

[6]The IANA database is available at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

ISO/IEC 8473-1:1998, Information technology — Protocol for providing the connectionless-mode network service: Protocol specification.[7]

ISO/IEC 9899:2018, Information technology—Programming languages—C.

OMG® Unified Modeling Language™ (OMG UML®), Version 2.5, March 2015.[8, 9]

[7]ISO publications are available from the International Organization for Standardization (https://www.iso.org/) and the American National Standards Institute (https://www.ansi.org/). IEC publications are available from the International Electrotechnical Commission (https://www.iec.ch) and the American National Standards Institute (https://www.ansi.org/).

[8]OMG publications are available from the Object Management Group, at http://www.omg.org.

[9]OMG, UML, and Unified Modeling Language are either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.[10]

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

— end station
— frame

This standard makes use of the following terms defined in IEEE Std 802.1Q:

— Edge Control Protocol (ECP)

The following terms are specific to this standard:

applicant: One of the two roles (with registrar) an application instance can have with respect to a Portal. The applicant controls a database that the Link-local Registration Protocol (LRP) replicates to the registrar in the neighbor Portal.

application instance: An instance of a Link-local Registration Protocol (LRP) application running in a particular Native or Proxy system.

Controlled system: An end system or relay system that has one or more target ports, but not their associated Portals.

Edge Control Protocol (ECP): A protocol that provides reliable delivery of control service data units (SDUs). (From IEEE Std 802.1Q)

Edge Control Protocol Data Unit (ECPDU): The basic unit of information transmitted or received by the Edge Control Protocol.

end station: A functional unit in an IEEE 802 network that acts as a source of, and/or destination for, link layer data traffic carried on the network. (From IEEE Std 802)

end system: A system attached to a network that is an initial source or a final destination of data transmitted across that network.

NOTE—The term “end system” is often used in this document in places where the reader of IEEE 802 standards would expect the term “end station,” in order to avoid confusion caused by standards relating to routers. For example, a router, as defined by IETF, is an IEEE 802 “end station,” but not an “end system,” because it is not typically an initial source or final destination. Where this standard specifically refers to the use of IEEE 802 services, the term “end station” is used. Where it refers to more generalized instances of connectionless services, the term “end system” is used.

frame: The format of aggregated bits from a medium access control (MAC) sublayer entity that are transmitted together in time. (From IEEE Std 802)

local target port: A target port within the same Native system as an application instance using the port, or a target port within a Controlled system serviced by an application instance in a Proxy system using the port.

[10]*IEEE Standards Dictionary Online* is available at https://dictionary.ieee.org. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

LRP application: A protocol, identified by an application ID, that makes use of the Link-local Registration Protocol (LRP).

Native system: An end system or relay system that has one or more target ports and their associated Portals.

Neighbor Portal: The one Portal to which a Portal has been connected.

neighbor target port: A target port in a system directly connected to a local port.

port: An entity defined by a {Chassis ID, Port ID} pair that is unique over the union of systems interconnected by the Link-layer Registration Protocol (LRP). A port can be tied to a Service Access Port (SAP) providing access to an instance of the MAC Service described in IEEE Std 802.1AC.

NOTE—See IEEE Std 802.1AC™ [B3].[11]

Portal: An instantiation of the Portal interface, along with instances of the applicant and/or registrar state machines and databases, associated with a single application instance, residing in a Native or Proxy system, and associated with a target port.

Proxy system: A system that has Portals, but not the associated target ports.

record: A subset of a database that is transferred as a single unit from the applicant to the registrar by the Link-local Registration Protocol (LRP).

registrar: One of the two roles (with applicant) an application instance can have with respect to a Portal. The registrar receives the copy of the database that the Link-local Registration Protocol (LRP) replicates from the applicant in the neighbor Portal.

relay system: A router or a bridge.

NOTE—The term "relay system" is often used in this document in places where the reader of IEEE 802 standards would expect the term "bridge." A relay system can, in theory, be a router, a bridge, or any other device that forwards a frame from one port to another, such that the frame is, in some useful sense, identifiable to other systems as being the same frame. Where this standard specifically refers to one or the other, the terms "router" or "bridge" are used. Where it refers to more generalized instances of connectionless services, the term "relay system" is used.

system: An end system or a relay system.

target port: A port in a Native or Controlled system with which a Portal is associated.

[11]The numbers in brackets correspond to those of the bibliography in Annex D.

4. Abbreviations

This standard contains the following abbreviations:

CID	Company ID
ECP	Edge Control Protocol
ECPDU	Edge Control Protocol Data Unit
LLDP	Link Layer Discovery Protocol (IEEE Std 802.1AB)
LLDPDU	Link Layer Discovery Protocol Data Unit
LRP	Link-local Registration Protocol
LRP-DS	Link-local Registration Protocol–Database Synchronization
LRP-DT	Link-local Registration Protocol–Data Transport
LRPDU	Link-local Registration Protocol Data Units
OUI	Organizationally Unique Identifier
TCP	Transmission Control Protocol
TLV	Type Length Value
UML	Unified Modeling Language

5. Conformance

5.1 Introduction

This clause specifies the mandatory and optional behaviors of conformant implementations of this standard. Following a subclause on Requirements terminology (5.2), four types of systems are defined in this clause:

a) The behaviors for Native end systems are specified in 5.3, 5.4 (mandatory) and 5.5 (optional).
b) The behaviors for Native relay systems are specified in 5.3, 5.6 (mandatory) and 5.7 (optional).
c) The behaviors for Proxy systems are specified in 5.3, 5.8 (mandatory) and 5.9 (optional).
d) The optional behaviors for Controlled systems are specified in 5.10.

5.2 Requirements terminology

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

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

The Protocol Implementation Conformance Statement (PICS) proformas (see Annex A) reflect the occurrences of the words "shall," "may," and "should" within the standard.

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

5.3 Common Native and Proxy required behaviors

A Native or Proxy system conformant to this standard shall:

NOTE—The specification of a particular LRP application can impose further constraints on LRP behavior.

a) Provide the LRP-DT instance state machine, as specified in 7.2 and 7.3;
b) Provide the Portal association maintenance state machines, as specified in 8.2;
c) Provide at least one of the Link-local Registration Protocol Data Transport mechanisms, Edge Control Protocol (ECP) or Transmission Control Protocol (TCP), as specified in 6.9.1 and 6.9.2;
d) Conform to the specifications for the format and encoding of Link-local Registration Protocol Data Units (LRPDUs) in Clause 9; and
e) Provide access to the managed objects required for the functions in item a in this subclause, according to the specifications in Clause 11.

5.4 Native end system required behaviors

A Native end system conformant to this standard shall, on one or more ports:

a) Conform to the required common capabilities specified in 5.3; and
b) Provide either the Applicant state machine as specified in 8.3, the Registrar state machines, as specified in 8.4, or both.

5.5 Native end system optional behaviors

A Native end system conformant to this standard that provides the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) should:

a) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for the LRP TCP Discovery TLV specified in C.2.2.

A Native end system conformant to this standard that does not provide the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) may:

b) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for LLDP TLVs specified in Annex C.

A Native end system conformant to this standard may:

c) Provide access to the managed objects required by 5.3 and 5.4 or selected from 5.5 using the YANG modules specified in Clause 12; and/or
d) Provide access to the managed objects required by 5.3 and 5.4 or selected from 5.5 using the MIB modules specified in Clause 13.

NOTE—The specification of a particular LRP application can make mandatory any of the optional behaviors specified in this clause (see B.11).

5.6 Native relay system required behaviors

A Native relay system conformant to this standard shall, on more than one port:

a) Conform to the required common capabilities specified in 5.3; and
b) Provide all of the Applicant and Registrar state machines, as specified in 8.3 and 8.4.

5.7 Native relay system optional behaviors

A Native relay system conformant to this standard that provides the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) should:

a) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for the LRP TCP Discovery TLV specified in C.2.2.

A Native relay system conformant to this standard that does not provide the TCP Link-local Registration Protocol Data Transport mechanism (item c in 5.3) may:

b) Conform to the required capabilities specified in IEEE Std 802.1AB and to the provisions for LLDP TLVs specified in Annex C.

A Native relay system conformant to this standard may:

c) Provide access to the managed objects required by 5.3 and 5.6 or selected from 5.7 using the YANG modules specified in Clause 12; and/or
d) Provide access to the managed objects required by 5.3 and 5.6 or selected from 5.7 using the MIB modules specified in Clause 13.

NOTE—The specification of a particular LRP application can make mandatory any of the optional behaviors specified in this clause (see B.11).

5.8 Proxy system required behaviors

A Proxy system conformant to this standard shall:

a) Provide the LRP-DT instance state machine, as specified in 7.2 and 7.3;
b) Provide the Portal association maintenance state machines, as specified in 8.2;
c) Provide all of the Applicant and Registrar state machines, as specified in 8.3 and 8.4;
d) Provide the LRP-DT TCP mechanism, as specified in 6.9.2;
e) Conform to the specifications for the format and encoding of LRPDUs in Clause 9; and
f) Provide access to the managed objects required for these functions, according to the specifications in Clause 11.

5.9 Proxy system optional behaviors

A Proxy system conformant to this standard may:

a) Provide access to the managed objects required by 5.3 and 5.8 using the YANG modules specified in Clause 12; and/or
b) Provide access to the managed objects required by 5.3 and 5.8 using the MIB modules specified in Clause 13.

NOTE—The specification of a particular LRP application can make mandatory any of the optional behaviors specified in this clause (see B.11).

5.10 Controlled system optional behaviors

A Controlled system, whether an end system or a relay system, should:

a) Conform to the required behaviors of IEEE Std 802.1AB Station and Media Access Control Connectivity Discovery, and to the provisions for LLDP TLVs specified in Annex C;
b) Provide access to the managed objects required to configure the address(es) of its Proxy system specified in 11.6.1; and/or
c) Provide access to the managed objects for monitoring LLDP TLVs as specified in 11.6.2.

A Controlled system, whether an end system or a relay system, may:

d) Provide access to the implemented managed objects using the YANG modules specified in Clause 12; and/or
e) Provide access to the implemented managed objects using the MIB modules specified in Clause 13.

6. Link-local Registration Protocol

6.1 Introduction

The Link-local Registration Protocol (LRP) is described in the following subclauses:

a) An overview of the LRP is given in 6.2.
b) Objectives and non-objectives are given in 6.3.
c) Descriptions of Native, Proxy, and Controlled systems are given in 6.4.
d) The role of the Link Layer Discovery Protocol (LLDP) is given in 6.5.
e) A description of Exploratory Hello LRPDUs is given in 6.6.
f) The handling of port state changes are given in 6.7.
g) The structure of the LRP databases is given in 6.8.
h) Two methods for connecting Portals with LRP Data Transport protocol (LRP-DT) are given in 6.9.
i) A description of the LRP Database Synchronization protocol (LRP-DS), which runs over LRP-DT to actually synchronize the applicant and registrar databases, is given in 6.10.
j) A brief description of the process of creating the LRP state machines is given in 6.11.
k) Security considerations are given in 6.12.

6.2 Overview

A brief explanation of the purpose of LRP is given in 1.7. This overview describes the relationships among the various functions comprising LRP in more detail. The basic protocol relationships are shown in Figure 6-1.

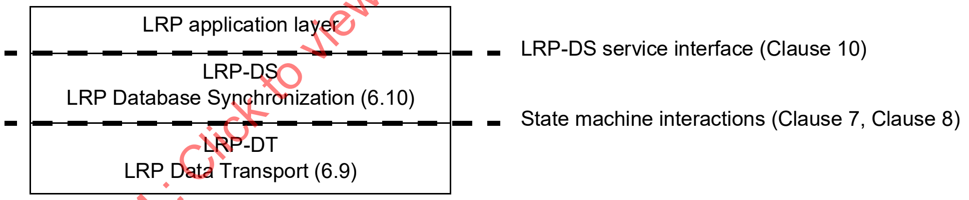

LRP application layer
LRP-DS service interface (Clause 10)
LRP-DS
LRP Database Synchronization (6.10)
State machine interactions (Clause 7, Clause 8)
LRP-DT
LRP Data Transport (6.9)

Figure 6-1—LRP protocol stack in a relay system

In Figure 6-1, the LRP application layer is the user of LRP. LRP is split into two layers: The LRP-DS database synchronization mechanism, and the LRP-DT data transport mechanisms. An application instance, operating at the LRP application layer, and a Portal operating at the LRP-DS layer, cooperate to manage applicant and registrar databases associated with target ports, as shown in Figure 1-1 and Figure 6-2. The LRP-DS Portal uses an LRP-DT instance to connect, via another system's LRP-DT instance, to a neighbor LRP-DS Portal. LRP-DT can use either the ECP or TCP to transport data. The state machines in the LRP-DS layer use the LRP-DT instances to exchange LRPDUs, by which means they copy the databases from system to system.

LRP-DT delivers data over a point-to-point association with the following services:

a) Pacing of transmissions to avoid overflowing the receiver's buffers.
b) Acknowledgment and retransmission of data to give some level of assurance it has been received.
c) In-order delivery of transmissions.

d) Repacking LRPDUs into suitably sized transmission units (packets) (TCP only).

e) Notification of a failure of a connection (failure to receive an acknowledgment) (TCP only).

Because LRP-DT provides these services, LRP-DS does not have a mechanism to retransmit data. Because ECP does not provide connection failure indications (item e, above) LRP-DS performs a periodic check (see 6.10.1) on the integrity of the database.

NOTE—Periodic integrity checks are performed, whether ECP or TCP is used for the LRP-DT.

This model is expanded in Figure 6-2, which shows the applicant and registrar databases, two systems A and B, each supporting two ports and the application instances of two different LRP applications, with both systems connected to other systems (off page to the right).

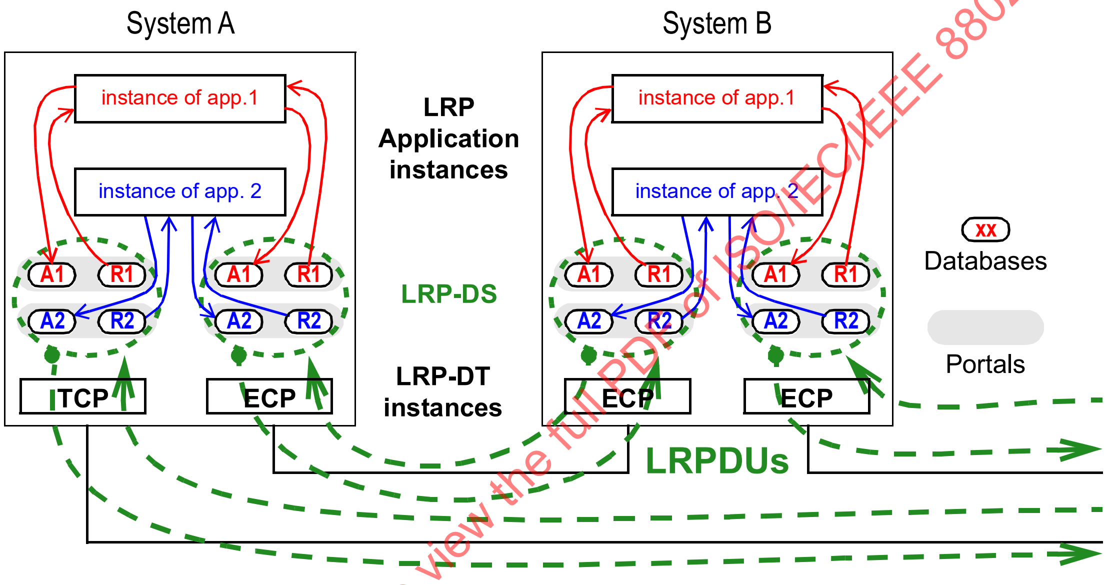


Figure 6-2—LRP application instances, databases, Portals, and LRP

6.3 Objectives and non-objectives

6.3.1 Objectives of LRP

The objectives of LRP are to:

a) Serve application instances as described in 6.2.

b) Remain independent of the syntax and semantics of the LRP application data contained in the applicant and registrar databases.

c) Provide a facility for an application instance to discover its peers, and to make and break associations between application instances.

d) Support multiple options for the transport mechanisms used to carry LRPDUs, in order to provide a range of capability/complexity trade-offs.

e) Transfer the application instances' data quickly, reliably, and efficiently with respect to bandwidth.

f) Efficiently transfer changes to parts of a database, without retransmitting the whole database.

g) Serve applicant and registrar databases whose size, per port, is on the order of magnitude of 1 Mbyte.

h) Support only point-to-point associations between application instances.
i) Support multiple point-to-point associations on one port.
j) By the use of TCP as an LRP-DT transport mechanism, provide for placing the application instance and Portal in a separate Proxy system from its target port, in order to facilitate central control of a network.
k) Support proxying for a Controlled system without requiring any new behaviors of the Controlled system, in order to make it possible to deploy a new LRP application without implementing an application instance in every system.
l) Make efficient use of TCP connections.
m) Support resource-constrained end systems.

6.3.2 Non-objectives of LRP

Certain possible objectives for LRP are explicitly excluded from the present document; if such features are required for a given LRP application, they must be provided by the LRP application specification, as follows:

a) Support point-to-multipoint or multipoint-to-multipoint associations among Portals.
b) Provide, in LRP, features for maintaining consistency across the various parts of a single database, e.g., locking all or a part of the database.

6.4 Proxy systems, Controlled systems, and target ports

Every Portal and its associated applicant and registrar databases are associated with a single target port. A target port can be associated with more than one Portal if those Portals serve different LRP applications, or if the Portals connect to different neighbor Portals. A target port provides access to a link that connects to one or more other target ports, typically in other systems.

There are three classes of systems relevant to this standard. The class into which a system falls is defined per LRP application; a system can be in one class for one LRP application, and in another class for another LRP application.

Native	A system in which the application instances, the target ports, and the Portals all reside.
Controlled	A system in which the target ports, but not the application instances and Portals, reside.
Proxy	A system in which the application instances and Portals, but not the target ports, reside.

The following term is also useful:

not-controlled Either a Native or a Proxy system. That is, one in which at least the application instances and Portals reside.

Each applicant and registrar database pair shown, for example, in Figure 1-1 or Figure 6-2, is associated with a specific Portal and target port. In order to support the objective item j in 6.3.1, this standard provides for either or both of the systems connected by the link to off-load the application instance and Portal to a separate Proxy system. The system offloading its application instance and Portal is called a *Controlled system*.

NOTE 1—This standard does not specify any means for off-loading some, but not all, of the Portals of an LRP application in a single Controlled system to a Proxy system.

NOTE 2—A system can be simultaneously a Proxy system and a Native system for the same LRP application, if it has Portals for its own target ports and also proxies for at least one other system.

NOTE 3—The term "system" is not synonymous with "chassis" or "box"; a physical device can contain parts or all of any number of systems.

Figure 6-3 illustrates two Native systems, A and B. Each Portal resides in the same system as its target port. Note that TCP could be used in place of ECP, in Figure 6-3.

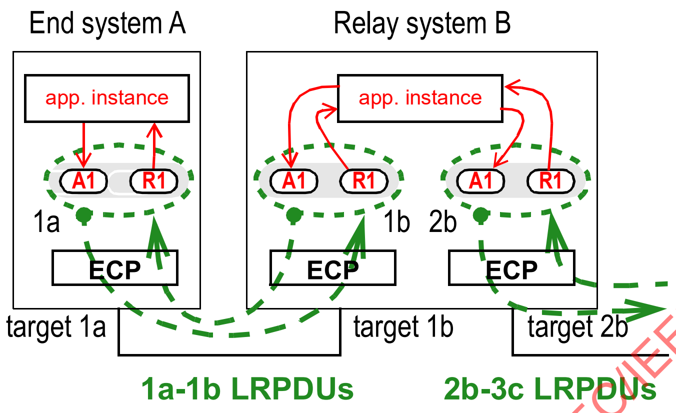


Figure 6-3—Native systems

In Figure 6-4, end system A from Figure 6-3 has been changed from a Native system to a Controlled system, and its Proxy system P has been added to the figure. As a consequence, system B must use TCP, not ECP, for its LRP-DT layer. The paths of the LRPDUs are shown passing "over the top" to Proxy system P, because they are not restricted to passing through the target port. Typically, running an application instance in a Proxy system results in some change of state in the Controlled system, and vice versa. For example, a resource reservation protocol might have to adjust parameters in the queue shapers on a target port. This kind of interaction is of concern to the LRP application specification, not to this standard. It is shown in Figure 6-4 and Figure 6-5 as an arrow labeled "other control." An existing system with no knowledge of an LRP application can be controlled by a Proxy system (item k in 6.3.1) so that it behaves as if it did understand the LRP application. All that is necessary is that the Controlled system be manageable by the Proxy system, perhaps via managed objects.

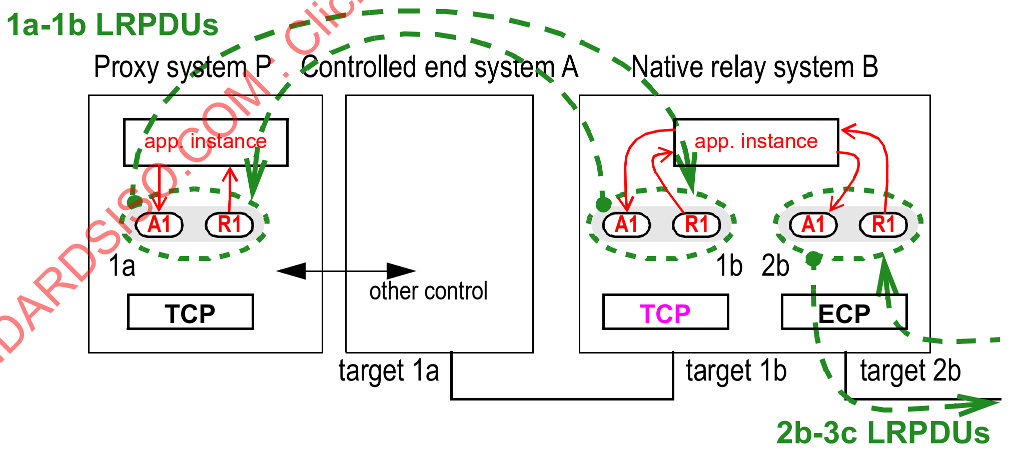


Figure 6-4—Controlled end system, Native relay systems

In Figure 6-5, we see what happens if relay system B of Figure 6-4 becomes a Controlled system to Proxy system Q. Now, the two Proxy systems exchange LRPDUs for the link between target ports 1a and 1b.

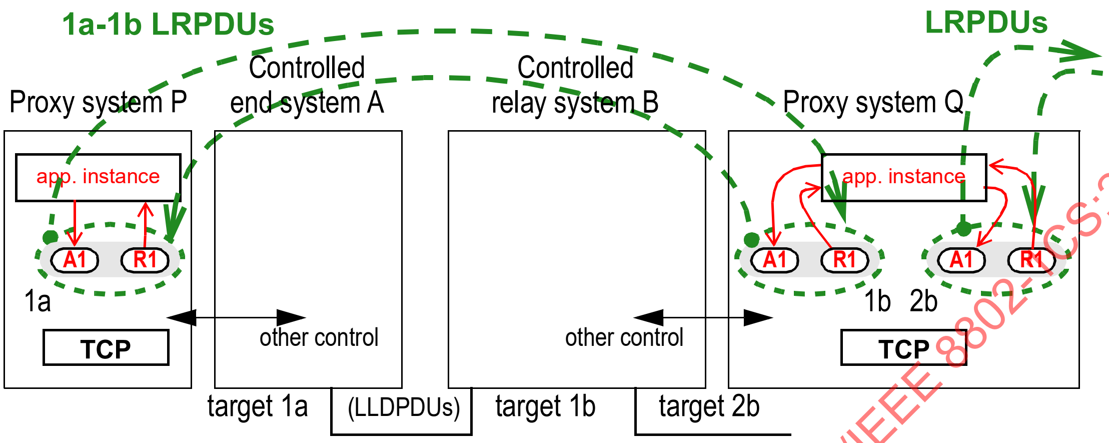


Figure 6-5—Controlled end and relay systems

As long as an LRP system uses TCP, rather than ECP, it is of no consequence to LRP or to an application instance in that system, whether its partner Portal is proxied or not. This makes it possible to design an LRP application that can be hosted either on independent cooperating relay systems, or on a centralized network controller system. Some useful examples of the use of Proxy/Controlled systems include:

a) **Proxied end systems:** A Proxy system can proxy for a relatively simple end system, and communicate to the neighboring relay system, so that the end system does not have to implement the LRP application.

b) **Proxied relay systems:** All relay systems in a network can use a central Proxy system. Then, a Native end system can participate in what, to it, appears to be a hop-by-hop LRP application protocol that works throughout the network, when in fact, the central Proxy system is a network controller that performs the LRP application's work in all of the relay systems in a single processor, communicating to the end systems via TCP.

c) **End systems' proxy and relay systems' proxy**: One system can proxy for a set of cooperating Controlled end systems, while another system is a proxy for the Controlled relay systems. The two proxies can interact efficiently on behalf of their respective sets of Controlled systems. This is illustrated in Figure 6-6. In this case, no LRPDUs pass between relay systems; all such traffic is internal to the network systems' proxy.

NOTE 4—TCP makes it possible to create Proxy and Controlled systems by masking whether a Native or a Proxy is exchanging LRPDUs with another Native or a Proxy system. This standard does not require, and does not specify, any particular communication between the Proxy and Controlled systems.

The choice to off-load an LRP application and its Portals to a Proxy system is made independently for each LRP application. A single system can be a Controlled system for one LRP application, a Native system for another LRP application, and a Proxy system for a third LRP application.

The specification of an LRP application can choose whether to allow Proxy systems or not. If, for example, the information passed in an LRP application could prevent a system from communicating with an adjacent Controlled system's Proxy system via TCP, then support of Proxy systems by that LRP application might be inadvisable.

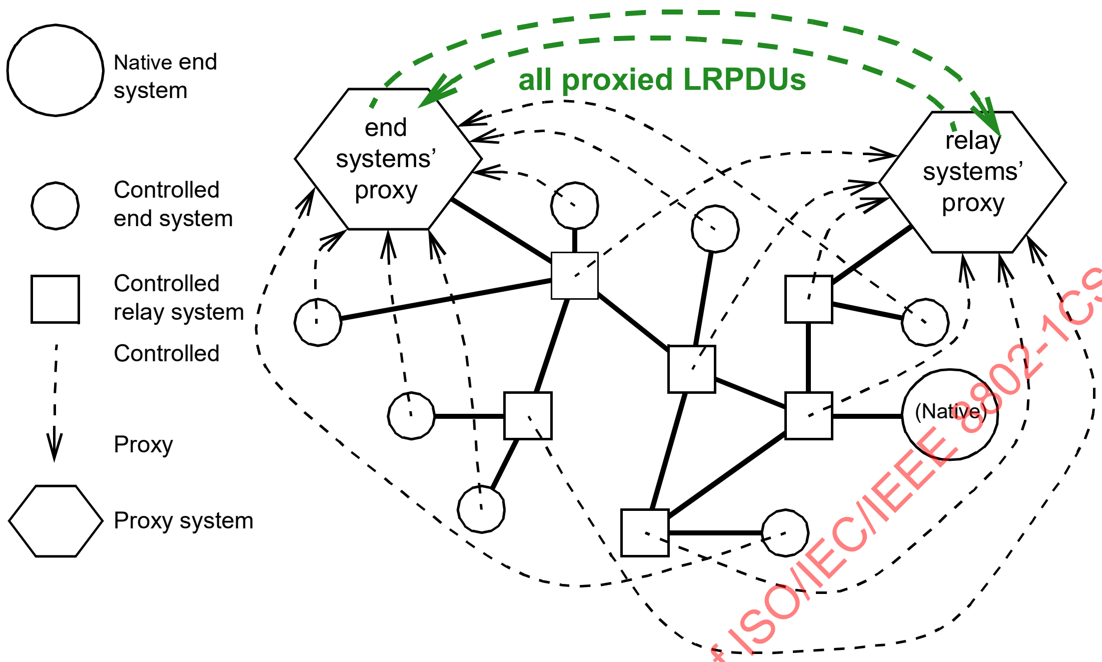


Figure 6-6—Communicating Proxy systems

6.5 Link Layer Discovery Protocol and target ports

The IEEE Std 802.1AB Link Layer Discovery Protocol (LLDP) can be used by systems to advertise and discover Portals by means of the LRP ECP Discovery TLV (C.2.1) and the LRP TCP Discovery TLV (C.2.2). When used, the LLDPDUs are sent and received through, and apply to, target ports, so only a Native or a Controlled system pass these TLVs. (A Proxy system has no target ports.) In Figure 6-5, for example, systems A and B are shown trading LLDPDUs.

If a Controlled system runs LLDP, then its Proxy system needs to access and/or control the LRP ECP Discovery TLVs and LRP TCP Discovery TLVs sent and received on the Controlled system's target ports. The means for coordinating this information is not specified in this standard, but see IEEE Std 802.1AB for a description of managed objects that make this possible.

Alternatively, the information that would be supplied via LLDP running either in a Native or a Controlled system can be configured statically in a Native system or in the Controlled's Proxy system. The managed objects required to configure this information are not defined in this standard, but could be defined in a particular application specification. (See B.3.) In addition, Exploratory Hello LRPDUs (6.6) can be used to simultaneously detect an associate with a neighboring system.

However the information about potential neighbor target ports is obtained, whether through a Native or Controlled system running LLDP, or through configuration of a Native or Proxy system, an application instance can use the Local Target Port request and Neighbor Target Port request primitives (10.2.2, 10.2.3) to convey that information to LRP.

NOTE—Running and monitoring LLDP requires more of an implementation than simply configuring the information in each system. However, running LLDP is much safer than configuration, because miswiring or misconfiguration can result in passing information between Portals about target port connections that do not, in fact, exist.

The Proxy/Controlled mechanisms defined in this standard require a meaningful way to identify the Controlleds’ target ports to the Proxy systems, even though IP allows the Proxy and Controlled systems to be in separate broadcast domains. The Chassis ID + Port ID mechanism of IEEE Std 802.1AB is used to identify individual target ports. An LRP application running in a network is not expected to work correctly if two or more target ports have the same Chassis ID and Port ID. It is up to those administrating the network(s) using LRP to ensure that target port identifiers are unique over an adequately large domain to meet the needs of the LRP applications deployed.

6.6 Exploratory Hello LRPDUs

In some situations, support for resource-constrained systems (item m in 6.3.1) leads to a desire to not run IEEE 802.1AB LLDP in the system, but to still have some means of discovering one’s neighbor target ports. The state machines in Clause 7 and Clause 8 permit the transmission and reception of an exploratory Hello LRPDU (9.4.2). This is a Hello LRPDU that carries neither the Neighbor Chassis ID TLV (9.4.2.8) nor the Neighbor Port ID TLV (9.4.2.9); it does advertise its own local target port with the My Chassis ID TLV (9.4.2.6) and the My Port ID TLV (9.4.2.7). The reception and transmission of exploratory Hello LRPDUs are enabled separately (see 7.2.2.1). By this means, the system transmitting the exploratory Hello LRPDUs does not need to acquire the chassis and port ID of its neighbor target port before transmitting the Hello LRPDU, either by configuration or by running LLDP.

Exploratory Hello LRPDUs can only be transmitted over ECP, not over TCP, because with ECP, there is no difficulty in determining to which local target port a received exploratory Hello LRPDU is directed; a target port can have only one instance of ECP, and ECP can have only one neighbor port.

6.7 Target port state

Since each LRP Portal and its databases are tied to a specific target port, it is expected that the state of the target port, e.g., whether or not the target port has an operational physical connection, will be important to the LRP application. However, different LRP applications can have different requirements for maintaining or altering the databases and/or propagating information to the rest of the network when a target port momentarily goes operational or non-operational. Furthermore, a Proxy system has only second-hand knowledge of the state of its Controlled system’s target ports. Therefore, the state of the target port has no direct impact on the LRP state machines described in Clause 7 and Clause 8.

The impact of the target port's state on the state machines can be different in different circumstances:

a) The LRP-DT ECP mechanism (6.9.1) uses the target port to carry LRPDUs, so a failed target port results in no LRPDUs being sent or received, which can cause Hello LRPDUs (9.4.2) to time out.
b) The LRP-DT TCP mechanism (6.9.2) can use the target port to carry LRPDUs. If so, a failed target port can cause the TCP connection (socket) to fail, which will destroy the LRP-DT instance and all its associated Portals. A failed target port can also cause Hello LRPDUs (9.4.2) to time out.
c) If the TCP connection does not pass through the target port, then a target port failure has no effect on the state machines in Clause 7 and Clause 8.
d) In any case, the failure of a port in any system in the network through which the TCP connection passes can cause an interruption in connectivity between Portals that can cause TCP connections to fail and/or Hello LRPDUs to time out.

It is up to the LRP application specification to define how to determine the state of the target port, how to convey that state to a Proxy system, and what actions to take when the state changes. The controls defined in Clause 10 allow an application instance both to receive indications of failures (Portal destruction, Hello LRPDU timeouts, or failure to synchronize the databases), and to command LRP to take action.

There are many kinds of failure of a target port; for example, a failed target port can support only one-way communication, it can have an excessive loss rate, or it can fail completely. There are many means to detect failures, including state reported by the MAC layer, the Internal Sublayer Service MAC_Operational parameter (11.2 of IEEE Std 802.1AC-2016 [B3]), or Connectivity Fault Management (Clause 18 of IEEE Std 802.1Q-2018). The best means can be different for different LRP applications; no particular means is specified in this standard.

6.8 LRP database structure

LRP imposes a small amount of structure upon the applicant and registrar databases. In particular:

a) An applicant or registrar database is divided into one or more records.
b) Each record has:
    1) An integer record identifier, unique over the records in the database;
    2) A sequence number. (The sequence number is used by LRP-DS only; it is not visible to the LRP application layer);
    3) A checksum computed over the whole record (9.4.6) that is computed by an LRP-DS Portal and exchanged in LRPDUs (The checksum is used by LRP-DS only; it is not visible to the LRP application layer); and
    4) One or more octets of data.
c) A given LRP-DT instance, using a given LRP-DT transport mechanism, can impose a maximum record size on an LRP application. The format of the LRPDU (Clause 9) limits a record's data to something less than 65 535 octets. If the LRP-DT ECP mechanism (6.9.1) is used, the maximum is further reduced to something less than the maximum frame size of the physical medium.
d) The interaction between LRP-DS and the LRP application layer, and between associated LRP-DS Portals, is defined in terms of adding, withdrawing, or altering whole records.
e) The data portion of a record is opaque to LRP; its structure and meaning are defined by the LRP application specification.

This division into records helps meet the goals of keeping LRP independent of the LRP applications it serves (item b in 6.3.1) and also being efficient when transferring parts of a database (item f in 6.3.1). The record identifier + sequence structure provides handles to identify the elements (records) that LRP exchanges with an application instance, so that LRP can replicate changes to the database, ensure its integrity across system restarts, and acknowledge the receipt of changes, without frequent transmissions of the entire database. An application instance can divide its applicant database into the parts that are expected to be (or have proven to be) relatively static from those that are dynamic, and put them in separate records. Then, LRP can transmit dynamic changes without affecting the static part of the database, or other dynamic records that are not changing at the same time, without understanding the details of the LRP application, and without itself implementing complex heuristics.

6.9 LRP-DT data transport mechanisms

6.9.1 LRP-DT ECP mechanism

LRPDUs can be transferred using the Edge Control Protocol (ECP) described in Clause 43 of IEEE Std 802.1Q-2018. ECP runs over the target port itself, so it can only be used by a Native system. One ECPDU (one frame) can carry a number of LRPDUs, but no LRPDU can span multiple ECPDUs. This limits the size of an LRPDU, and hence the size of one record. The reason for this limitation is that, unlike TCP (6.9.2), ECP fails silently if no acknowledgment is received after some number of retries (default 3, see 12.26.1.2 of IEEE Std 802.1Q-2018). If such a failure dropped one fragment of an LRPDU, then the next

LRPDU or LRPDU fragment could not be parsed by the receiver. Limiting an LRPDU to a single ECPDU ensures that only whole LRPDUs can be lost by ECP.

The ECP ackTimerInit value (D.2.12.6 of IEEE Std 802.1Q-2018), shall be 20.480 ms (RTE=11). This value is visible as a managed object (lrpAckTimerInit, 11.3.1).

NOTE—The choice made by ECP for a value of ackTimerInit, when there is more than one protocol using an ECP instance, is not fully defined by IEEE Std 802.1Q-2018.

Because ECP allows only a single instance of ECP per physical port, which instance converses with only one neighbor system, ECP can have multiple point-to-point associations (item i in 6.3.1) only if an ECP instance is used by more than one LRP application. The associations are multiplexed by a field in the LRPDUs (My Portal Number, 9.4.2.4) to distinguish each Portal's (association's) LRPDUs.

The application instance specifies the destination MAC address for transmitted frames carrying ECPDUs (item b:3:i in 10.2.3.1). If the LRP ECP Discovery TLV (C.2.1) is being transmitted, this address shall be the same as the destination MAC address used for the LLDPDUs. The Nearest Bridge group address, Nearest Customer Bridge group address, and Provider Bridge group address (Table 8-1, Table 8-2, or Table 8-3 in IEEE Std 802.1Q-2018) are reasonable choices.

6.9.2 LRP-DT TCP mechanism

LRPDUs can be transferred over a TCP connection. An LRPDU can contain up to 65 535 octets of data, the maximum value of the LRPDU's TLV Length field (9.3.3).

Because TCP can support any number of LRP-DS Portal associations through one LRP-DT instance, multiple point-to-point associations (item i in 6.3.1) are fully supported for a single port. The associations are multiplexed by a field in the LRPDUs (My Portal Number, 9.4.2.4) to efficiently utilize the TCP connection (item l in 6.3.1).

The interaction between LRP-DS and a TCP connection is described in terms of sockets, as defined by IEEE Std 1003.1. There are two ways to create a TCP socket using the functions defined in IEEE Std 1003.1: Active TCP OPEN (6.9.2.1) or Passive TCP OPEN (6.9.2.2).

6.9.2.1 Active TCP OPEN

A socket can be created using an active TCP OPEN with the following steps:

a) The IEEE 1003.1 `socket()` function creates a socket.

b) The IEEE 1003.1 `bind()` function can optionally attach a particular local IP address to the socket, if more than one IP address is available for use.

c) The IEEE 1003.1 `connect()` function creates a TCP connection to a particular remote system, for use by one LRP-DT instance.

6.9.2.2 Passive TCP OPEN

A socket can be created using a passive TCP OPEN with the following steps:

a) The IEEE 1003.1 `socket()` function creates a passive socket to be used for enabling the receipt of other systems' active TCP OPENs.

b) The IEEE 1003.1 `bind()` function is required to attach a local IP address to the socket. See 6.9.2.3 for the selection of the TCP port number parameter to be used in the IEEE 1003.1 `bind()` function.

c) The IEEE 1003.1 `listen()` function enables the socket to field `connect()` functions issued by other systems.

These first three steps are executed once for each local IP address used by a system for the LRP-DT TCP mechanism. The socket created is not associated with any particular LRP-DT instance. The last step is performed once for each LRP-DT instance.

d) The IEEE 1003.1 `accept()` function clones a new socket to pair with another system's `connect()` function, leaving the original passive socket available for further `accept()` functions.

6.9.2.3 LRP TCP port number

There is no fixed value for the TCP port number for LRP. The TCP port number and IP address(es) for a particular Proxy system can be advertised in the LRP TCP Discovery TLV (C.2.2) and passed to LRP from the application instance through the Neighbor Target Port request (item b:4:iii in 10.2.3.1).

NOTE—The means by which a TCP Port number is selected is not specified by this standard. When a Native system issues the IEEE 1003.1 `bind()` function, it can then advertise that local TCP Port number (selected from the Dynamic Port range specified by IETF RFC 6335 [B15]) in its LRP TCP Discovery TLVs. A Proxy system can install the selected TCP Port numbers into the LRP TCP Discovery TLVs transmitted by its Controlled systems using managed objects (lldpV2ConfigLrpTcpControlledTcpPortNumber in 13.5.3). If a fixed TCP port number for LRP is assigned, that number can be advertised.

6.9.3 Quality of Service considerations

LRP is expected to be used to carry resource reservation protocols on systems implementing the queuing mechanisms required for demanding real-time uses. If frames carrying LRPDUs bypass these queuing mechanisms, that are required to provide such Quality of Service, then the real-time data streams can be disrupted.

NOTE—This standard does not specify a particular priority value (11.1 of IEEE Std 802.1AC-2016 [B3]) or a DSCP value (IETF RFC 2474 [B8]) for the ECP or TCP frames or packets. Requirements vary, but a high-priority best-effort class of service is usually a good choice for frames or packets carrying LRPDUs.

6.10 LRP-DS database synchronization mechanism

6.10.1 LRP-DS Protocol operation

The use of records by LRP-DS is modeled after the use of Link State Advertisements (LSAs) in ISO/IEC 10589:2002 IS-IS [B19]. Four LRPDUs are carried between Portals to convey database information:[12]

1) **Hello LRPDU:** Establishes and maintains an LRP-DS association between LRP Portals.
2) **Record LRPDU:** A record, with record identifier, sequence number, checksum, and data.
3) **Partial List LRPDU:** A list of a subset of the {record identifier, sequence number, checksum} triplets (no data) in the database.
4) **Complete List LRPDU:** A complete list of all of the {record identifier, sequence number, checksum} triplets (no data) in the database.

A Record LRPDU is sent from the applicant to the registrar to add, change, or delete a record. A Partial List LRPDU is sent from the registrar to the applicant to acknowledge receipt of one or more Record LRPDUs.

[12]There is a fifth "Stop LRPDU" defined in 9.4.1, but this is merely a syntactic convention for padding an ECPDU to meet minimum frame length requirements on some physical media.

A Complete List LRPDU is sent periodically by the registrar to check whether the registrar database matches the applicant database.

See Clause 8 for descriptions of the state machines that transmit and receive these LRPDUs, and Clause 9 for their formats.

NOTE—The checksum on the data contained in Record LRPDUs is not computed for data integrity; LRP-DS runs over a transport protocol, either ECP or TCP, that supplies reasonable assurance that LRPDUs are delivered without being corrupted. The checksum is used to catch errors that can happen when one or both connected system are reinitialized, when two records can be transmitted with the same record identifier and sequence number, but with different data.

6.10.2 LRP-DS database storage

Figure 6-7 illustrates the entire LRP-DS data path, showing the transformations in the data as it is replicated. The procedure described here is somewhat more elaborate than an implementation might use, but it illustrates the underlying principles behind the design of LRP. The only parts of Figure 6-7 that constrain an implementation are the externally visible LRPDUs. Striped and blocks are used to represent pieces of information that are likely to have only a momentary existence. The numbered steps in Figure 6-7 are as follows:

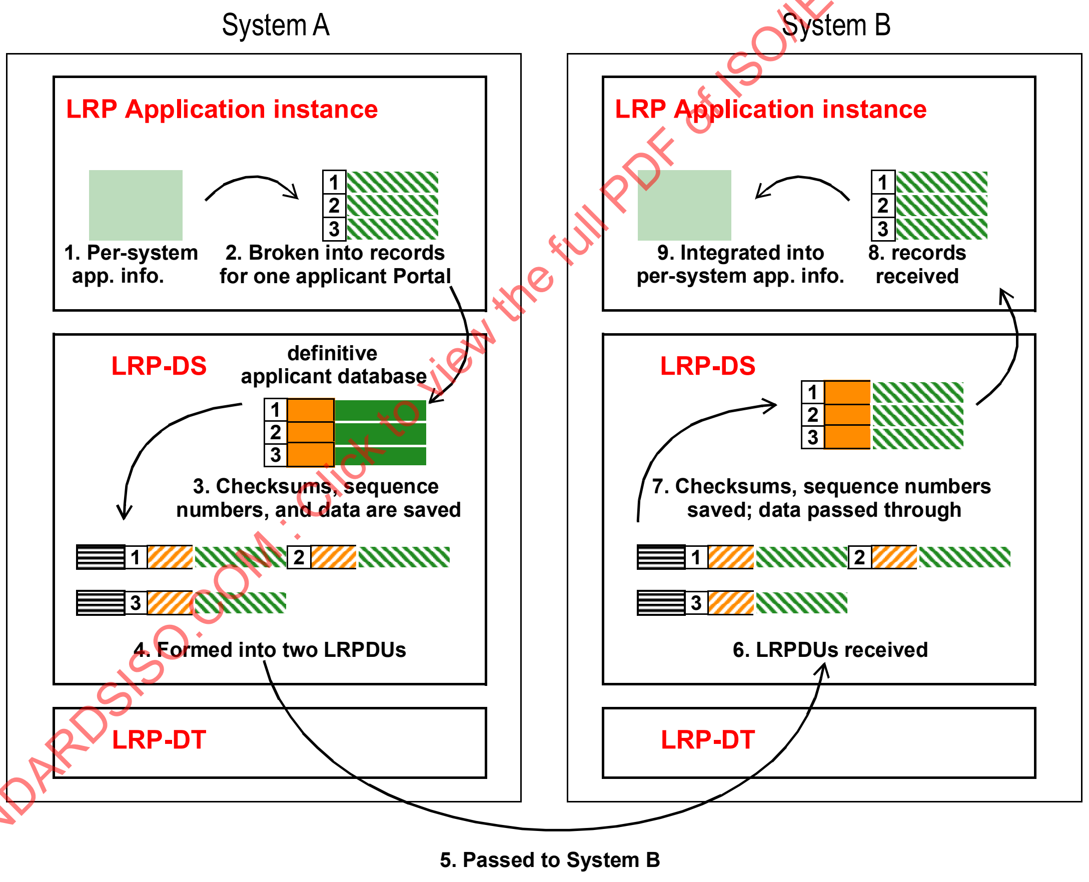

System A
System B
LRP Application instance
1. Per-system app. info.
2. Broken into records for one applicant Portal
LRP-DS
definitive applicant database
3. Checksums, sequence numbers, and data are saved
4. Formed into two LRPDUs
LRP-DT
5. Passed to System B
LRP Application instance
9. Integrated into per-system app. info.
8. records received
LRP-DS
7. Checksums, sequence numbers saved; data passed through
6. LRPDUs received
LRP-DT
1
2
3

Figure 6-7—LRP-DS data path

1) We can posit "Per-system LRP application information" that contains information related to an application instance's purpose.
2) We can further imagine that the application instance creates an image of an applicant database to be replicated to its partner on a particular port. From this image, the application instance creates a set of records, each with a record number, to be transmitted. The records may be a simple division of the image into sections, but see B.5 for another possibility.
3) The application instance uses the primitives in 10.3.1 to convey the records to the LRP-DS Portal, where the definitive copy of the applicant database resides. LRP-DS supplies a sequence number and checksum for each record.
4) As governed by the applicant state machines described in 8.3, LRP-DS collects the records that need to be transmitted into Record LRPDUs (9.4.3), and passes them to LRP-DT.
5) LRP-DT moves the Record LRPDUs from the applicant Portal to the registrar Portal, using acknowledgments, timeouts, and retries to deliver the data quickly and reliably.
6) Following the procedures described in 8.4, LRP-DS in the partner Portal receives the Record LRPDUs from LRP-DT and splits out the various records.
7) LRP-DS responds with Partial List LRPDUs as required, and saves the record numbers, sequence numbers, and checksums for each record. The actual data is not saved by LRP-DS, but is passed up to the application instance along with the record number and sequence number.
8) The application instance receives the numbered records (including deletions), and reconstructs the exact image of the applicant database (the registrar database), posited in item 2, above. This process may be as simple as concatenating the records in numerical order, but see B.5 for another possibility.
9) The application instance merges this registrar database into its per-system LRP application information, which may well trigger applicant-side operations on this or other Portals in the system.

As a practical matter, a given implementation might merge the LRP application with LRP-DS, or might use a completely different set of primitives from those in Clause 10, but the specification of an LRP application is most easily described in terms of the Clause 10 primitives.

6.11 State machine creation

The LRP-DS and LRP-DT state machines can reside either in a Native system, with the target ports, or in a Proxy system, separate from the Controlled systems' target ports (6.4). LRP-DS and LRP-DT state machines are created based on information supplied through the Local Target Port request and Neighbor Target Port request primitives (10.2.2, 10.2.3). The table imTargetPortList (7.2.2.1) stores this information. Changes to the imTargetPortList trigger the creation and destruction of LRP-DT state machines (7.3), which in turn lead to the creation and deletion of Portals (Clause 8). Figure 6-8 illustrates these relationships and shows the routines in Clause 7 that manage these tables and state machines.

Input to LRP from the application instance is required to actually create the LRP state machines and start their operation. As shown in Figure 6-8, LRP uses the imTargetPortList (7.2.2.1) to control the creation of ECP and TCP LRP-DT instances, as well as passive TCP sockets (6.9.2.2) to receive TCP connections from other systems. The receipt of TCP connection requests from other systems can create TCP LRP-DT instances. Also, ECP LRP-DT instances can be created directly by ProcessLocalTargetRequest (7.2.3.1) for the reception of Exploratory Hello LRPDUs (6.6).

Each LRP-DT instance is one end of an LRP-DT connection to exactly one neighbor LRP-DT instance. One LRP-DT instance can be associated with one or more different LRP applications for one or more local target ports, each {target port, LRP application} pair constituting a Portal (Clause 8), which implements the LRP-DS sublayer.

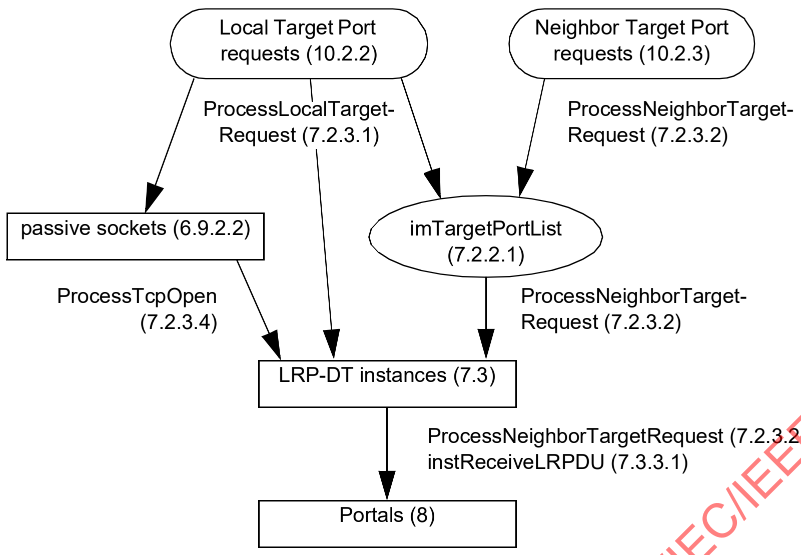

Local Target Port requests (10.2.2)
Neighbor Target Port requests (10.2.3)
ProcessLocalTarget-Request (7.2.3.1)
ProcessNeighborTarget-Request (7.2.3.2)
passive sockets (6.9.2.2)
imTargetPortList (7.2.2.1)
ProcessTcpOpen (7.2.3.4)
ProcessNeighborTarget-Request (7.2.3.2)
LRP-DT instances (7.3)
ProcessNeighborTargetRequest (7.2.3.2), instReceiveLRPDU (7.3.3.1)
Portals (8)

Figure 6-8—State machine creation

A Portal and its state machines is created either with the creation of an LRP-DT instance, or after the receipt of the first Hello LRPDU (9.4.2), from a neighbor Portal.

A Portal transmits and receives LRPDUs through its associated LRP-DT instance. The Portals paired with a given LRP-DT instance share the transport connection to the neighbor LRP-DT instance.

6.12 Security considerations

LRP uses a transport layer, either ECP or TCP. ISO/IEC 10589:2002 IS-IS [B19] commonly uses security measures described in IETF RFC 5304 [B12]. However, since IS-IS does not use an underlying transport protocol, these measures are not of much use to LRP. Similarly, IETF RFC 7987 [B16] does not apply to LRP, because there is no Lifetime field to protect, although a Lifetime field could be useful to a particular LRP application.

The ECP protocol can use MACsec (IEEE Std 802.1AE™ [B4]) to provide a level of security for LRP. Because every data LRPDU carries a sequence number, replay of data or list LRPDUs is not a problem over either ECP or LRP. Replay of Hello LRPDUs could disrupt making a connection, but TCP prevents that problem. ECP has no replay protection for Hellos, but MACsec does prevent replays.

IPsec (IETF RFC 4301 [B11]) or Transport Layer Security (TLS, IETF RFC 8446 [B18]) can be used to further secure TCP connections. The use of these protocols is not specified in this standard.

7. LRP-DT Procedures

7.1 Introduction

This clause describes the following procedures and state machines:

a) The mechanisms used to create and destroy LRP-DT instances are described in 7.2.
b) The operation of an LRP-DT instance is described in 7.3.

7.2 LRP-DT instance maintenance

7.2.1 LRP-DT instance maintenance and Portal creation overview

LRP-DT instances are instantiations of the LRP-DT instance state machine (7.3.4), and exist only in a Native or a Proxy system. They are created and destroyed by the following methods:

a) Local Target Port requests (10.2.2) issued by the application instance call ProcessLocalTargetRequest (7.2.3.1) to create and destroy descriptions of local target ports, that is, target ports residing in either that same Native system, or in a Controlled system controlled by the Proxy system in which the application instance resides.
b) Similarly, Neighbor Target Port requests (10.2.3) issued by the application instance call ProcessNeighborTargetRequest (7.2.3.2) to create or destroy descriptions of neighbor target ports, each of which is attached to a specific local target port.
c) If a local target port is described, but no neighbor target port is described, then
    1) If ECP is enabled (item b:3 in 10.2.2.1), then ProcessLocalTargetRequest creates an ECP LRP-DT instance, but no Portal.
    2) If a TCP address (and Port number) is supplied, then ProcessLocalTargetRequest creates a Passive TCP OPEN socket (6.9.2.2) if not already created, but no LRP-DT instance or Portal is created, yet.
d) When a neighbor target port is described and attached to the local target port description, then:
    1) If the local and neighbor target ports have ECP addresses supplied, then ProcessNeighborTargetRequest creates a Portal and attaches it to the LRP-DT instance.
    2) If the local and neighbor target ports have TCP addresses (and TCP port number) supplied, then ProcessNeighborTargetRequest creates both a TCP LRP-DT instance and a Portal, which it attaches to the LRP-DT instance. It issues an Active TCP OPEN (6.9.2.1) on the LRP-DT instance, if one does not already exist.
e) When an IEEE 1003.1 `accept()` call succeeds on a socket created in item c:2, above, ProcessTcpOpen (7.2.3.4) creates a new TCP LRP-DT instance, but not a Portal.
f) When a Hello LRPDU (9.4.2) is received on an LRP-DT instance for an unknown neighbor target port, then First Hello indication (10.2.5) and Associate Portal request (10.2.4) are used to allow or disallow the Portal to associate with its neighbor.
g) Local Target Port requests, Neighbor Target Port requests, and TCP connection failures can trigger the destruction of target port descriptions, which in turn, destroy the associated Portals and LRP-DT instances.
h) Duplicated TCP connections can cause duplicated TCP LRP-DT instances. Subsequent operation of the Receive Hello state machine (8.2.5) can cause one of the duplicated LRP-DT instances and its TCP socket to be destroyed.

Only TCP LRP-DT instances need to establish connections with active or passive OPEN operations. ECP has no connection establishment operations.

The details of this process are described in the following subclauses.

7.2.2 Instance maintenance variables

7.2.2.1 imTargetPortList

An imTargetPortList resides in each Native or Proxy system. It maintains a list of all local and neighbor target ports of interest to the system. Entries in the imTargetPortList are created or destroyed by the ProcessLocalTargetRequest routine (7.2.3.1). Each entry in the list is associated with a single application instance and local target port, and thus has a unique value for the triple {imPplAppId, imPplMyChassisId, imPplMyPortId}.

In turn, each entry in the imTargetPortList has an attached list (imPplNeighborList, item g, below) of neighboring target ports. Entries in the imPplNeighborList are created or destroyed by the ProcessNeighborTargetRequest (7.2.3.2) routine. Each entry in the imPplNeighborList has, in conjunction with its parent imTargetPortList entry, a unique value for the 5-tuple {imPplAppId, imPplMyChassisId, imPplMyPortId, imPplNbrChassisId, imPplNbrPortId}.

The imTargetPortList is, in effect, a list of known target ports for which Portals are expected to be created. For each expected Portal, imTargetPortList has local and remote address information to be used for an LRP-DT instance to carry that Portal's LRPDUs. Note that this is not a list of all possible Portals, as Portals can also be created as a result of receiving Hello LRPDUs (item f in 7.2.1).

The relationship of the information in the imTargetPortList to that in the LRP ECP Discovery TLV and LRP TCP Discovery TLV (C.2.1, C.2.2) carried by IEEE 802.1AB LLDP is governed by the application instance via the primitives in Clause 10. See 6.5.

Each entry in the imTargetPortList contains the following items:

a) **imPplAppId:** The AppId of an LRP application (9.2).

b) **imPplMyChassisId:** The Chassis ID (9.4.2.6) of the Native or Controlled system's target port (8.5.2 of IEEE Std 802.1AB-2016).

c) **imPplMyPortId:** The Port ID (9.4.2.7) of the Native or Controlled system's target port (8.5.3 of IEEE Std 802.1AB-2016).

d) **imPplMyEcpEnable:** A Boolean value indicating whether this target port accepts LRPDUs transmitted over ECP.

e) **imPplMyTcpInfo:** The LRP TCP IP address and TCP port number information for the Native or Controlled system's target port (C.2.2.6). This item can be empty.

NOTE 1—The procedures in 7.2.3.1 ensure that either imPplMyEcpEnable is TRUE, imPplMyTcpInfo is not empty, or that both conditions are met.

f) **imPplHelloParms:** Hello parameters to be used by any generated Hello state machines (item c in 10.2.2.1).

g) **imPplNeighborList:** A list, possible empty, of neighbor target ports. Each entry in the list contains:

   1) **imPplNbrChassisId:** The Chassis ID (9.4.2.8) the neighbor system's target port (8.5.2 of IEEE Std 802.1AB-2016).

   2) **imPplNbrPortId:** The Port ID (9.4.2.9) a neighbor system's target port (8.5.3 of IEEE Std 802.1AB-2016).

3) **imPplNbrEcpInfo:** The LRP ECP MAC address for a neighbor system's target port (C.2.1). This item can be empty.
4) **imPplExploreXmit:** A Boolean value indicating that the local Portal controlled by this imPplNeighborList entry can transmit Exploratory Hello LRPDUs (6.6).
5) **imPplExploreRecv:** A Boolean value indicating that the local Portal controlled by this imPplNeighborList entry is willing to receive Exploratory Hello LRPDUs (6.6).
6) **imPplNbrTcpInfo:** The LRP TCP IP address and TCP port number information for a neighbor system's target port (C.2.2.6). This item can be empty.

NOTE 2 —The procedures in 7.2.3.1 ensure that imPplNbrEcpInfo and imPplNbrTcpInfo are never both empty.

7) **imPplActiveOpen:** A Boolean value indicating that this system needs to issue an active TCP connection OPEN operation (6.9.2.1) for this target port pair.
8) **imPplPortalRef:** A reference (1.5) to the Portal associated with this neighbor target port, or null, if none.

A system shall not allow the creation of an imTargetPortList such that any of the following are true:

h) Two or more different imPplNeighborList entries have the same values of the 5-tuple {imPplAppId, imPplMyChassisId, imPplMyPortId, imPplNbrChassisId, imPplNbrPortId}.
i) In one imPplNeighborList entry, imPplNbrEcpInfo is null and either imPplExploreXmit or imPplExploreRecv is TRUE.

7.2.3 Instance maintenance routines

7.2.3.1 ProcessLocalTargetRequest

In a Native or Proxy system, a Local Target Port request (10.2.2) results in a call to ProcessLocalTargetRequest to create or destroy an entry in the imTargetPortList (7.2.2.1), the list of {LRP application, target port} pairs in the system. This routine takes the parameters given in the Local Target Port request as inputs. ProcessLocalTargetRequest then modifies the imTargetPortList as follows:

a) If the parameters of the Local Target Port request are invalid (e.g., specify a non-existent target port), theProcessLocalTargetRequest terminates and returns an error code for the Local Target Port request (item c:1 in 10.2.2.2).
b) If no entry exists in the imTargetPortList with imPplAppId, imPplMyChassisId, and imPplMyPortId values matching the AppId, Chassis ID, and Port ID values specified in the Local Target Port request (item a:1, item b:1, and item b:2 in 10.2.2.1), then:
   1) If ECP is disabled and no IP address information is supplied in the Local Target Port request, the request is ignored and ProcessLocalTargetRequest terminates.
   2) Otherwise (ECP enabled and/or IP address is supplied), ProcessLocalTargetRequest creates a new entry in the imTargetPortList, and fills the imPplMyEcpEnable, imPplMyTcpInfo, and imPplHelloParms values in the new imTargetPortList entry from the information specified in the Local Target Port request (item b:3, item b:4, item c:2 in 10.2.2.1). The imPplNeighborList item in the new entry is empty. Processing continues (item d, below).
c) Otherwise (a matching entry is found):
   1) If ECP is disabled and no IP address information is supplied in the Local Target Port request, the existing imTargetPortList entry is deleted. Processing continues (item d, below).
   2) Otherwise (ECP enabled and/or IP address is supplied), If the imPplMyEcpEnable, imPplMyTcpInfo, and imPplHelloParms values in the imTargetPortList entry match the information specified in the Local Target Port request (item b:3, item b:4, item c:2 in 10.2.2.1),

request is ignored and ProcessLocalTargetRequest terminates. If they do not match, an error is reported (item c:1 in 10.2.2.2) and ProcessLocalTargetRequest terminates.

d) If an entry in the imTargetPortList is created with a value for an IPv4 and/or an IPv6 address in imPplMyTcpInfo (item e in 7.2.2.1) that was not previously in the imTargetPortList, then ProcessLocalTargetRequest creates a new socket and issues an IEEE 1003.1 `listen()` call to initiate a passive TCP OPEN for each new IP address (6.9.2.2).

e) If item c:1, above, resulted in the deletion if an imTargetPortList entry, then ProcessLocalTargetRequest performs calls ProcessNeighborTargetRequest (7.2.3.2) on each entry in the imPplNeighborList (item g in 7.2.2.1) of the imTargetPortList entry to delete the neighbors and their Portals. These actions can cause the destruction of LRP-DT instances.

f) If any deleted imTargetPortList entry was the only remaining entry in the list to specify a particular IP address, then the socket on which the IEEE 1003.1 `listen()` function was issued for passive TCP OPENs for that IP address is destroyed with the IEEE 1003.1 `shutdown()` call.

7.2.3.2 ProcessNeighborTargetRequest

Whenever a Neighbor Target Port request (10.2.3) is made in a Native or Proxy system, the system calls ProcessNeighborTargetRequest to create or destroy an LRP-DT instance and Portal for a specific neighbor target port. ProcessNeighborTargetRequest is also called by the Receive Hello state machine (8.2.5) when a Portal Association is approved by the application instance using the Associate Portal request (10.2.4). ProcessNeighborTargetRequest takes the parameters given in the Neighbor Target Port request as inputs in the former case, and takes the fields in the received Hello LRPDU (9.4.2) in the latter case. The routine then modifies the imPplNeighborList (item g in 7.2.2.1) in the imTargetPortList entry specified in the Neighbor Target Port request (7.2.2.1, item c:1 in 10.2.3.1) as follows:

a) If the parameters of the Neighbor Target Port request are invalid (e.g., specify a non-existent imTargetPortList entry), ProcessNeighborTargetRequest terminates and returns an error code for the Neighbor Target Port request (item b:1 in 10.2.3.2).

b) If no entry exists in the selected imPplNeighborList with imPplAppId, imPplNbrChassisId, and imPplNbrPortId values (item a, item h:1, item g:2 in 7.2.2.1) matching the AppId, Chassis ID, and Port ID values specified in the Neighbor Target Port request (item e:1 in 10.2.2.1, item b:1 and item b:2 in 10.2.3.1), then:

    1) If no ECP address or IP address information is supplied in the Neighbor Target Port request, the request is ignored and ProcessNeighborTargetRequest terminates.

    2) Otherwise (ECP address and/or IP is supplied), ProcessNeighborTargetRequest creates a new entry in the imPplNeighborList, and fills the imPplNbrEcpInfo, imPplExploreXmit, imPplExploreRecv and imPplNbrTcpInfo values in the new imPplNeighborList entry from the ECP address and IP address information specified in the Neighbor Target Port request (item b:3, item b:4 in 10.2.3.1). ProcessNeighborTargetRequest sets imPplActiveOpen to FALSE and imPplPortalRef to null (item h:7, item g:8 in 7.2.2.1). Processing continues (item d, below).

c) Otherwise (a matching entry is found):

    1) If no ECP address or IP address information is supplied in the Neighbor Target Port request, then instDestroyPortal (7.3.3.5) is called to delete the portal referenced in the imPplPortalRef (item g:8 in 7.2.2.1) of the deleted entry, if any. The imPplNeighborList entry is then deleted. ProcessNeighborTargetRequest terminates.

    2) Otherwise (ECP address and/or IP is supplied), if the imPplNbrEcpInfo and imPplNbrTcpInfo values in the imPplNeighborList entry do not match the ECP address and IP address information specified in the Neighbor Target Port request (item b:3, item b:4 in 10.2.3.1), an error is reported (item b:1 in 10.2.3.2) and ProcessNeighborTargetRequest terminates.

    3) Otherwise (imPplNbrEcpInfo and imPplNbrTcpInfo values in the imPplNeighborList entry do match the ECP address and IP address values specified in the Neighbor Target Port request),

then ProcessNeighborTargetRequest examines all LRP-DT instances. If a TCP LRP-DT instance is found that lists the same imTargetPortList entry in its instTargetPortList (7.3.2.8) that was given in the Neighbor Target Port request, and has the same IP addresses as the one returned by PickIpAddresses, then the request is ignored and ProcessNeighborTargetRequest terminates. Otherwise, the imPplNeighborList entry is selected for further processing.

NOTE—Because LRP-DT reuses TCP connections in order to conserve resources (item d:1, below), a connection that would have been selected as an active TCP connection can make use of an existing LRP-DT instance's passive TCP connection. If that connection is lost, the application instance can issue a new Neighbor Target Port request (10.2.3) for an already-extant imPplNeighborList, whereupon ProcessNeighborTargetRequest will initiate a new (active) TCP connection, if necessary. This is the reason that this last bullet (item c:3) can cause a new LRP-DT instance to be created for an already-extant imPplNeighborList entry that does not have one.

d) At this point, a new imPplNeighborList entry has been created, or an existing entry has been selected for which there is no LRP-DT instance. PickIpAddresses (7.2.3.3) is called with imPplMyTcpInfo and imPplNbrTcpInfo (item e and item g:6 in 7.2.2.1). If PickIpAddresses indicates success, and also indicates that an active TCP connection is to be made, then:
    1) ProcessNeighborTargetRequest examines all LRP-DT instances. If an LRP-DT instance is found that lists the same imTargetPortList entry in its instTargetPortList (7.3.2.8) that was given in the Neighbor Target Port request, and has the same IP addresses as the one returned by PickIpAddresses, then that LRP-DT instance is selected for further processing by ProcessNeighborTargetRequest.
    2) Otherwise, (no matching LRP-DT instance was found), then a new LRP-DT instance is created by calling instCreateInstance (7.3.3.2) with the imPplNeighborList entry information and the information returned by PickIpAddresses. This LRP-DT instance is selected for further processing by ProcessNeighborTargetRequest. instActiveTcp (7.3.2.1) is set to TRUE. Normal operation of the newly-created LRP-DT instance state machine (7.3.4) will trigger an Active TCP OPEN (6.9.2.1).
e) If PickIpAddresses succeeded, but indicated that no active TCP is to be made, then ProcessNeighborTargetRequest terminates processing successfully.
f) If PickIpAddresses failed, then the imPplMyEcpEnable and imPplNbrEcpInfo (item d and item g:3 in 7.2.2.1) are examined. If neither is empty, then an ECP LRP-DT instance is created, by calling instCreateInstance (7.3.3.2), and selected for further processing by ProcessNeighborTargetRequest. If both are empty, a failure indication is returned by ProcessNeighborTargetRequest (item b:1 in 10.2.3.2).
g) Having selected or created an ECP or active TCP LRP-DT instance, ProcessNeighborTargetRequest calls instCreatePortal (7.3.3.4) to create a Portal, if one does not already exist. pamNeedHello is set TRUE for a TCP LRP-DT instance or an ECP LRP-DT instance whose imPplExploreXmit flag is TRUE, else sets pamNeedHello to FALSE. ProcessNeighborTargetRequest also adds a reference to the Portal to imPplPortalRef (item g:8 in 7.2.2.1) and to the LRP-DT instance's instPortalList (7.3.2.9).

7.2.3.3 PickIpAddresses

This routine takes as inputs two Application descriptor fields (C.2.2.6): imPplMyTcpInfo (item e in 7.2.2.1), representing the local (Native or Proxy system's) IP address information, and imPplNbrTcpInfo (item g:6 in 7.2.2.1), representing the remote system's IP address information. PickIpAddresses returns as output either a failure indication, or the following information:

a) A "this end" IP address,
b) An "other end" IP address (or "unknown"),
c) An indication of whether the addresses (a and b) are IPv4 or IPv6,
d) A Boolean value indicating that this system should perform an active TCP OPEN (6.9.2.1) as part of creating an LRP-DT instance.

PickIpAddresses operates as follows:

e) If both Application descriptors' Address info fields (C.2.2.6.2) indicate IPv6 addresses (addrIPv6), then the IPv6 addresses are returned.
f) Otherwise (one or the other of the address fields did not indicate an IPv6 address), if both Application descriptors' Address info fields (C.2.2.6.2) indicate IPv4 addresses (addrIPv4), then the IPv4 addresses are returned.
g) Otherwise (one Application descriptor indicates IPv4 only, and the other IPv6 only), a failure indication is output.

The Address info fields (C.2.2.6.2) of inputs are then examined, and a return value for the active TCP OPEN choice is taken from Table 7-1. (See B.7 for a discussion of the need for Table 7-1.)

Table 7-1—Active TCP OPEN required

	Neighbor's Active/Passive preference		
My Active/Passive preference	**apNoPref**	**apActive**	**apPassive**
apNoPref	TRUE	FALSE	TRUE
apActive	TRUE	TRUE	TRUE
apPassive	FALSE	FALSE	TRUE

The principles driving this table are that:

— Both sides always make a passive TCP OPEN (6.9.2.2).
— If the preferences gives a clear choice as to which makes an active OPEN (6.9.2.2), that choice is respected.
— If neither party has a preference, or if their preferences conflict, both systems make an active OPEN.

7.2.3.4 ProcessTcpOpen

ProcessTcpOpen runs continuously in a Native or Proxy system supporting the LRP-DT TCP mechanism (6.9.2). It issues an IEEE 1003.1 `accept()` on every passive OPEN socket created by ProcessLocalTargetRequest (7.2.3.1). Whenever an `accept()` call succeeds, indicating that a remote system has issued an active OPEN (6.9.2.2), ProcessTcpOpen creates a TCP LRP-DT instance (7.3) with instActiveTcp (7.3.2.1) set to FALSE and instNeighborAddress and instNeighborPortNumber (7.3.2.4, 7.3.2.5) set to the remote IP address obtained from the `accept()` call. The new LRP-DT instance's instPortalList (7.3.2.9) is empty; no Portal can be created until a Hello LRPDU is received (7.3.3.1).

NOTE—The operation of IEEE 1003.1 sockets ensures that the newly created LRP-DT instance will have a different pair of IP addresses and port numbers from any other TCP LRP-DT instance.

7.3 LRP-DT instance

7.3.1 LRP-DT instance overview

An LRP-DT instance is an instantiation of the LRP-DT instance state machine (7.3.4). Its purposes are to:

a) Establish a connection to another LRP-DT instance via TCP. (No connection is needed for ECP.)
b) Create and/or destroy Portals' state machines (8.2, 8.3, 8.4).
c) Receive LRPDUs and assign them to a Portal for processing.

An LRP-DT instance serves some number of target ports (7.3.2.8). An ECP LRP-DT instance runs in a Native system only, and sends and receives all of its LRPDUs through the single target port that it serves. A TCP LRP-DT instance runs in a Native or Proxy system, can serve any number of local target ports, and can send or receive its LRPDUs through any port, or different ports at different times.

7.3.2 LRP-DT instance variables

7.3.2.1 instActiveTcp

A Boolean value that is TRUE if and only if instMyAddress (7.3.2.2) and instNeighborAddress (7.3.2.4) are TCP addresses and this LRP-DT instance is using the active (6.9.2.1), not the passive (6.9.2.2), form of TCP OPEN.

7.3.2.2 instMyAddress

The address of the local system for this LRP-DT instance; the address used as a destination address by the neighbor LRP-DT instance. The address includes a type (MAC, IPv4, or IPv6) and an address of that type.

7.3.2.3 instMyPortNumber

The local port number for this TCP connection, or 0, if this connection uses ECP, instead of TCP.

7.3.2.4 instNeighborAddress

The address of the neighbor LRP-DT instance; the address used as a destination address by this LRP-DT instance. The address includes a type (MAC, IPv4, or IPv6) and an address of that type.

7.3.2.5 instNeighborPortNumber

The remote port number for this TCP connection, or 0, if this connection uses ECP, instead of TCP.

7.3.2.6 instReconnectTimer

An integer valued timer that decrements once every second, if and only if its value is non-zero.

7.3.2.7 instReconnectReset

An integer number of seconds to wait before trying to remake the next failed TCP connection.

7.3.2.8 instTargetPortList

A list of references to the entries in the imTargetPortList (7.2.2.1) attached to this LRP-DT instance. This is a list of target ports in the Native or Controlled system served by this LRP-DT instance.

7.3.2.9 instPortalList

A list of references to the Portals attached to this LRP-DT instance.

7.3.2.10 instKillInstance

A Boolean signal indicating that the LRP-DT instance state machine is to cease operation.

7.3.3 LRP-DT instance routines

7.3.3.1 instReceiveLRPDU

This routine is called whenever an LRPDU is received on an LRP-DT instance. The LRPDU is processed according to the following:

a) Stop LRPDUs (9.4.1) are discarded without further processing.
b) If the LRPDU type field (9.3.2) of the LRPDU is not one of the values typeHelloLRPDU, typeRecordLRPDU, typePartialListLRPDU, or typeCompleteListLRPDU from Table 9-3, the LRPDU is counted in instDiscardedLRPDUs (11.4.1) and discarded without further processing.
c) If a Partial List LRPDU (9.4.4), Complete List LRPDU (9.4.5), or Record LRPDU (9.4.3) is received, instReceiveLRPDU searches the list of Portals in the LRP-DT instance’s instPortalList (7.3.2.9) for a Portal whose pamNeighborPortalNumberValid (8.2.2.6) is TRUE, and whose pamNeighborPortalNumber (8.2.2.7) matches the My Portal Number field (9.4.2.4) of the received LRPDU. If none is found, the LRPDU is counted in instDiscardedLRPDUs (11.4.1) and discarded without further processing. If found, then the LRPDU is presented to the Portal using actReceivePartialList (8.3.3.3), actReceiveCompleteList (8.3.3.4), or regReceiveWriteRecord (8.4.4.1), according to the LRPDU type, and instReceiveLRPDU terminates. This standard does not specify the action to be taken by instReceiveLRPDU if more than one of its attached Portals has the same value for pamNeighborPortalNumber.
d) Otherwise (a Hello LRPDU, 9.4.2, is received), instReceiveLRPDU checks to see whether or not both of the Neighbor Chassis ID TLV and Neighbor Port ID TLV are present in Hello LRPDU, and proceeds as follows:
    1) If both TLVs are missing, the LRP-DT instance is an ECP LRP-DT instance, there is only one imTargetPortList entry attached to the LRP-DT instance (instTargetPortList, 7.3.2.8), and that entry’s imPplNeighborList contains an entry with the imPplExploreRecv flag = TRUE (item g:5 in 7.2.2.1), then an exploratory Hello LRPDU (6.6) has been received. Processing of the Hello LRPDU proceeds with item e, below.
    2) Otherwise (if any of the conditions of item d:1 are not met), instReceiveLRPDU scans just those imTargetPortList entries listed in the LRP-DT instance’s instTargetPortList (7.3.2.8) for an entry whose imPplAppId, imPplMyChassisId, and imPplMyPortId fields (item a, item b, and item c in 7.2.2.1) match the appId field, Neighbor Chassis ID TLV, and Neighbor Port ID TLV of the received Hello LRPDU (9.4.2.1, 9.4.2.8, 9.4.2.9). If a matching entry is found, then processing proceeds with item e, below.
    3) Otherwise (no match is found), this LRP-DT instance does not support the target port named by the Hello LRPDU; the LRPDU is counted in instDiscardedLRPDUs (11.4.1) and discarded, and no further processing takes place.
e) At this point, we know that the LRP-DT instance serves the local target port named (except item d:1, above) in the Hello LRPDU. A Portal to process this Hello LRPDU is either found or created, in the following manner:
    1) The selected imTargetPortList’s imPplNeighborList (item g in 7.2.2.1) is searched for an entry whose imPplNbrChassisId and imPplNbrPortId values (item h:1, item g:2 in 7.2.2.1) match the My Chassis ID TLV and My Port ID TLV, respectively, from the Hello LRPDU. In this search, an imPplNeighborList entry with the imPplExploreRecv flag set and null values for imPplNbrChassisId and imPplNbrPortId is deemed to match any Hello LRPDU. This enables the receipt of an exploratory Hello LRPDU (6.6). If a match is found, then the Portal named in

the imPplPortalRef value (item g:8 in 7.2.2.1) is selected, and processing continues with item f, below.

2) Otherwise, (the Hello LRPDU matches some target port served by this LRP-DT instance, but not any particular Portal), all of the Portals attached to some LRP-DT instance, but *not* this LRP-DT instance, are examined to see if any match the AppId, My Chassis ID TLV, My Port ID TLV, Neighbor Chassis ID TLV, and Neighbor Port ID TLV from the Hello LRPDU. If none is found, then this was a Hello LRPDU for a new Portal. instReceiveLRPDU creates a new Portal by calling instCreatePortal (7.3.3.4) with pamNeedHello=FALSE, and processing continues (item f, below).

NOTE—Because no two entries in the imTargetPortList (7.2.2.1) have the same values for imPplMyChassisId and imPplMyPortId, any Portal found by this search is unique, and is attached to an LRP-DT instance that serves the same target port as the one on which the Hello LRPDU was received. Since an ECP LRP-DT instance cannot be attached to two different target ports (7.2.3.1), a matching Portal can be found by item e:2 only for two TCP LRP-DT instances. The semantics of IEEE 1003.1 sockets ensure that one of these is an Active TCP OPEN (instActiveTcp = TRUE, 7.3.2.1), and one a Passive TCP OPEN (6.9.2.1, 6.9.2.2), and also that the socket at one end of a TCP connection is active, and the other is passive.

3) Otherwise (the Hello LRPDU matches some Portal attached to another TCP LRP-DT instance), there are two TCP LRP-DT instances handling the same Portal association. The My Chassis ID TLV and My Port ID TLV of the received Hello LRPDU are compared to the imPplMyChassisId and imPplMyPortId (item b, item c in 7.2.2.1) of the target port to decide which LRP-DT instance "wins." The comparison is made by first concatenating the Chassis ID and Port ID of the receiving target port together as a single octet string, Chassis ID first, doing the same with the neighbor target port's Chassis ID and Port ID (from the Hello LRPDU) and comparing the two resultant strings as unsigned octets, in lexical order. If the receiving system's Chassis ID and Port ID are earlier in lexical order, and the receiving LRP-DT instance's instActiveTcp variable (7.3.2.1) is TRUE, then the existing Portal's LRP-DT instance wins, and the Hello LRPDU is discarded and counted in instDiscardedLRPDUs (11.4.1). Otherwise, the new Hello LRPDU wins, and the Portal is moved from the old TCP LRP-DT instance to the TCP LRP-DT instance on which the Hello LRPDU was received by altering the LRP-DT instances' instPortalLists (7.3.2.9). If an LRP-DT instance's instPortalList becomes null, instDestroyInstance (7.3.3.3) is called.

f) Having selected a Portal, instReceiveLRPDU does the following:
   1) If the Hello LRPDU was an exploratory Hello LRPDU (6.6) accepted under item d:1, above, and if imPplNbrChassisId and imPplNbrPortId are both null, then the My Chassis ID TLV and My Port ID TLV are copied to imPplNbrChassisId and imPplNbrPortId, respectively.
   2) Either way, instReceiveLRPDU then calls the Portal's pamReceiveHello (8.2.3.4) routine to process the Hello LRPDU.

7.3.3.2 instCreateInstance

instCreateInstance is called from ProcessNeighborTargetRequest (7.2.3.2) to create an LRP-DT instance and its state machines. Inputs to instCreateInstance are a reference to an imPplNeighborList entry (and by implication to the imTargetPortList entry to which it belongs, see 7.2.2.1), and the information returned to ProcessNeighborTargetRequest by PickIpAddresses (7.2.3.3). These inputs are used to initialize the LRP-DT instance's state machine variables (7.3.2).

7.3.3.3 instDestroyInstance

This routine examines an LRP-DT instance, and destroys it, if appropriate. If:

a) The instTargetPortList (7.3.2.8) is empty; or
b) If the instPortalList (7.3.2.9) is empty and instActiveTcp (7.3.2.1) is TRUE;

Then instDestroyInstance sets instKillInstance (7.3.2.10) to TRUE to bring the LRP-DT instance state machine (7.3.4) to a graceful halt. Otherwise, the LRP-DT instance, although it may be idle (have no attached Portals), is kept available to receive Hello LRPDUs and initiate new Portal associations via instReceiveLRPDU (7.3.3.1).

7.3.3.4 instCreatePortal

instCreatePortal instantiates a Portal, including its state machines (8.2, 8.3, 8.4). The inputs are:

a) A reference (1.5) to an LRP-DT instance;
b) A reference (1.5) to an entry in the imTargetPortList (7.2.2.1) specifying the local target port in the Native or Controlled system (“my” target port).
c) The Chassis ID of the neighbor target port or null.
d) The Port ID of the neighbor target port or null.
e) An indication of whether pamNeedHello (8.2.2.13) is to be initialized to TRUE or FALSE.

instCreatePortal inserts a reference (1.5) to the Portal into the LRP-DT instance’s instPortalList (7.3.2.9). It also fills the Portal state machines’ variables as shown in Table 7-2.

Table 7-2—Initialization of Portal state machine variables

State machine variable	Copy from	Reference	Initialize
pamAppId (8.2.2.1)	imPplAppId	item a in 7.2.2.1	
pamMyChassisId (8.2.2.2)	imPplMyChassisId	item g:1 in 7.2.2.1	
pamMyPortId (8.2.2.3)	imPplMyPortId	item c in 7.2.2.1	
pamNeighborChassisId (8.2.2.4)	input item c	item g:1 in 7.2.2.1	
pamNeighborPortId (8.2.2.4)	input item d	item g:1 in 7.2.2.1	
pamMyPortalNumber (8.2.2.9)			new[a]
pamNeighborPortalNumberValid (8.2.2.6)			FALSE
pamNeighborPortalNumber (8.2.2.7)			any
pamNeedHello (8.2.2.13)	input item e		
pamHelloTime (8.2.2.15)	imPplHelloParms	item f in 7.2.2.1	
pamMyAppInfo (8.2.2.16)	imPplHelloParms	item f in 7.2.2.1	

[a]A new Portal number is generated that is unique among the Portal numbers attached to the same LRP-DT instance.

Setting the pamNeedHello variable determines whether the new Portal’s Send Hello state machines (8.2.4) will attempt to transmit Hello LRPDUs immediately after the LRP-DT instance is operational (pamNeedHello = TRUE), or whether the Portal will wait to receive a Hello LRPDU and have the Portal association approved by the application instance before initiating Hello LRPDU transmissions (pamNeedHello = FALSE).

7.3.3.5 instDestroyPortal

Input is a reference (1.5) to a Portal. This routine sets the pamKill variable (8.2.2.23) to TRUE to terminate the Portal. This causes all of the Portal state machines to terminate gracefully. The Portal is removed from any LRP-DT instance that references it (instPortalList, 7.3.2.9), and all of the Portal's state machines and variables are destroyed. When finished, a Portal status indication (10.2.6) is sent to the application instance. If the LRP-DT instance's instPortalList is then empty, instDestroyInstance (7.3.3.3) is called.

7.3.4 LRP-DT instance state machine

The LRP-DT instance state machine is defined in Figure 7-1.

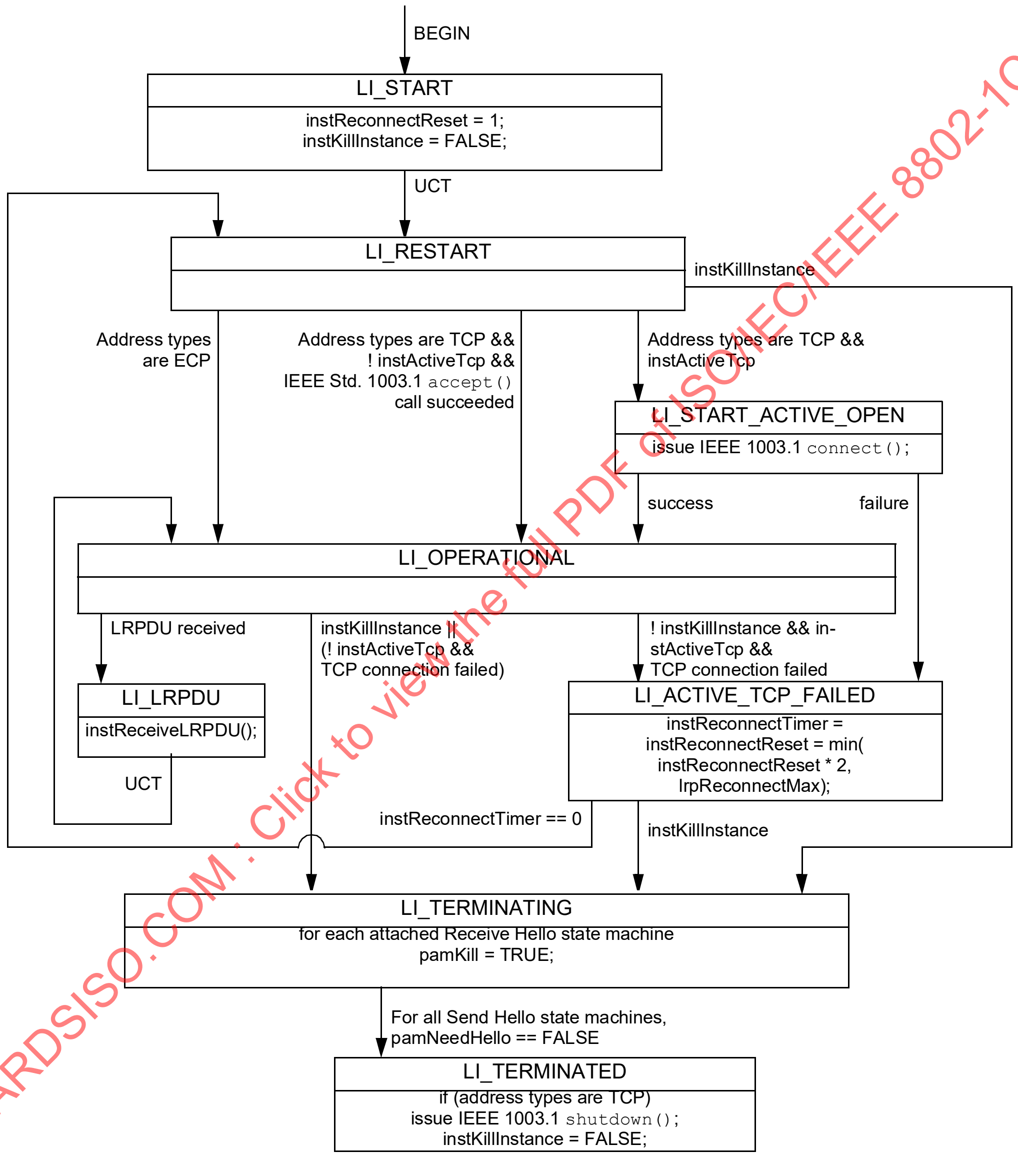

BEGIN
LI_START
instReconnectReset = 1;
instKillInstance = FALSE;
UCT
LI_RESTART
instKillInstance
Address types are ECP
Address types are TCP && ! instActiveTcp && IEEE Std. 1003.1 accept() call succeeded
Address types are TCP && instActiveTcp
LI_START_ACTIVE_OPEN
issue IEEE 1003.1 connect();
success
failure
LI_OPERATIONAL
LRPDU received
instKillInstance || (! instActiveTcp && TCP connection failed)
! instKillInstance && instActiveTcp && TCP connection failed
LI_LRPDU
instReceiveLRPDU();
UCT
LI_ACTIVE_TCP_FAILED
instReconnectTimer = instReconnectReset = min(instReconnectReset * 2, lrpReconnectMax);
instReconnectTimer == 0
instKillInstance
LI_TERMINATING
for each attached Receive Hello state machine
pamKill = TRUE;
For all Send Hello state machines, pamNeedHello == FALSE
LI_TERMINATED
if (address types are TCP)
issue IEEE 1003.1 shutdown();
instKillInstance = FALSE;

Figure 7-1—LRP-DT instance state machine

A passive TCP OPEN is initiated by ProcessLocalTargetRequest (7.2.3.1) before the LRP-DT instance state machine is created. ECP requires no OPEN procedure. Only an active TCP OPEN requires action by the LRP-DT instance state machine. A TCP connection is considered to have failed if an IEEE 1003.1 socket routine, e.g., `recv()` or `send()`, returns an error code of `ECONNRESET`, `ENOTCONN`, `ETIMEDOUT`, or `EPIPE`. The state machine retries an active TCP OPEN after a waiting period. The loss of a passive TCP

OPEN destroys the socket, the LRP-DT instance, and any attached Portals' Hello state machines (8.2). An ECP LRP-DT instance does nothing about the loss of a non-existent connection.

8. Portal

8.1 Introduction

This clause describes a Portal in the following subclauses:

a) The maintenance of associations between pairs of neighbor Portals by the exchange of Hello LRPDUs is described in 8.2.
b) The operation of the Applicant state machine, that transmits Record LRPDUs and receives Partial List LRPDUs, is described in 8.3.
c) The operation of the Partial list state machine that receives Record LRPDUs and responds with Partial List LRPDU, and the Complete list state machine that transmits Complete List LRPDUs are described in 8.4.

The relationships among the various state machines and major continuously running processes are illustrated in Figure 8-1.

8.2 Portal association maintenance

8.2.1 Portal association maintenance overview

An association between two Portals is established, maintained, and/or destroyed by the use of Hello LRPDUs (9.4.2), through the Portal association maintenance state machines defined in this Clause 8. The Portal state machines are created in two ways:

a) When explicit information about a neighbor target port is supplied in an imPplNeighborList entry in the imTargetPortList (7.2.2.1, 10.2.3), the LRP-DT instance and Portal are both created.
b) When only information about the local (Native or Controlled) target port is given in the imTargetPortList (10.2.2), an LRP-DT instance can be created (7.2), but the Portal is created only after a Hello LRPDU has been received.

The Portal's Send Hello state machines (8.2.4) transmit, and the Receive Hello state machine (8.2.5) receives, Hello LRPDUs. These messages confirm that the LRP-DT instances are connected correctly and then provide a slow keep-alive to ensure that the Portals are operational. These state machines support a three-way handshake that includes the application instance in the decision to create a Portal. An example of the exchange is illustrated in Figure 8-2.

When the first Hello LRPDU is received on a Portal (this same Hello LRPDU can have created the Portal), an indication (First Hello indication, 10.2.5) is sent to the application instance. The application instance responds with an Associate Portal request (10.2.4) specifying whether the Portal association is approved. If it is approved, then the Portal proceeds to operate, starting with the completion of the Hello LRPDU handshake.

In the case of item a, above, the identity of the neighboring target port is known, and the Portal commences the transmission of Hello LRPDUs as soon as the LRP-DT instance becomes operational. In the case of item b, the identity of the neighboring target port is not known until the receipt of the initial Hello LRPDU. In that case, transmission of Hello LRPDUs is not initiated until the Portal association is approved by the application instance.

The Applicant state machine and Partial list state machine are enabled to operate only after the Receive Hello state machine's pamMyHelloStatus variable (8.2.2.8) has reached the state hsConnected.

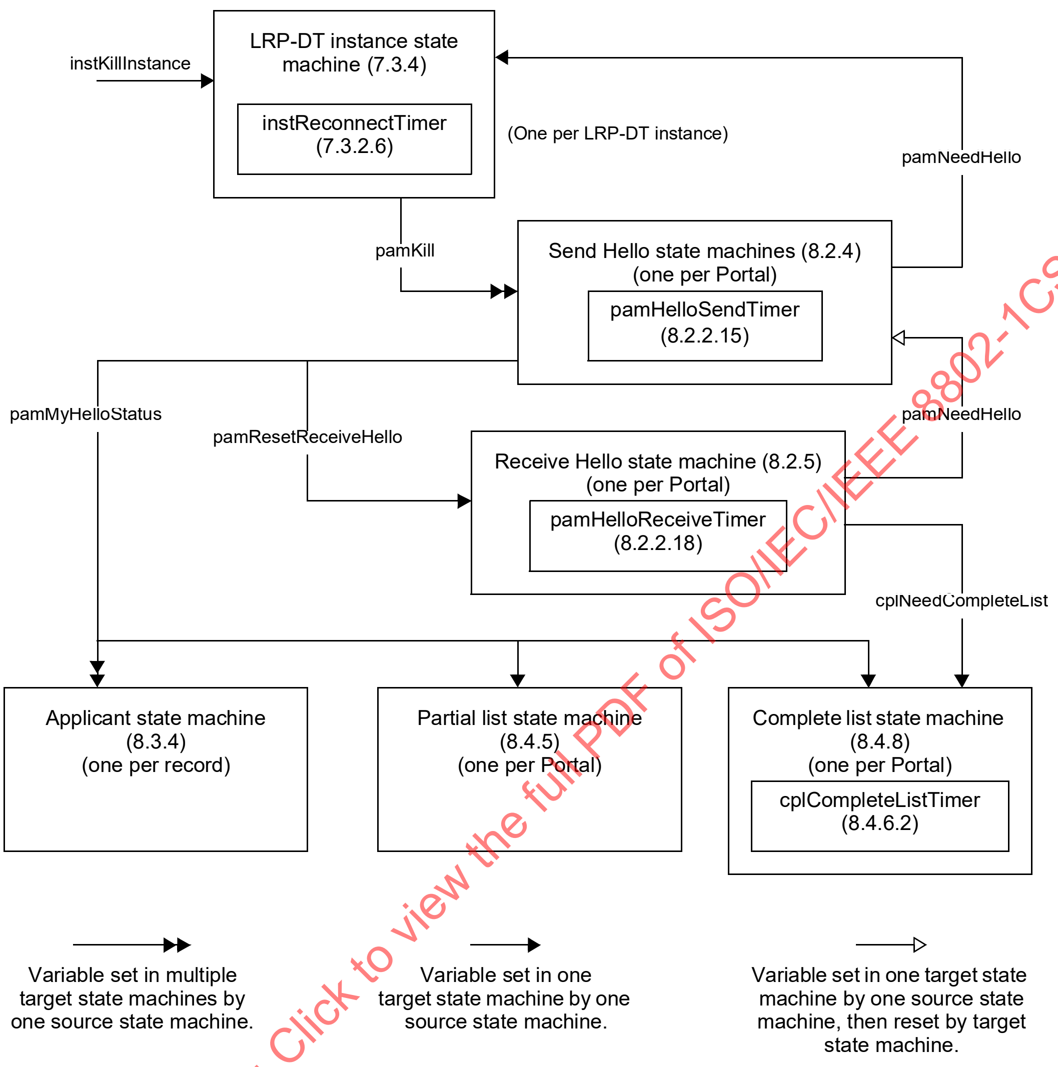

LRP-DT instance state machine (7.3.4)
instReconnectTimer (7.3.2.6)
instKillInstance
(One per LRP-DT instance)
pamNeedHello
pamKill
Send Hello state machines (8.2.4) (one per Portal)
pamHelloSendTimer (8.2.2.15)
pamMyHelloStatus
pamResetReceiveHello
pamNeedHello
Receive Hello state machine (8.2.5) (one per Portal)
pamHelloReceiveTimer (8.2.2.18)
cplNeedCompleteList
Applicant state machine (8.3.4) (one per record)
Partial list state machine (8.4.5) (one per Portal)
Complete list state machine (8.4.8) (one per Portal)
cplCompleteListTimer (8.4.6.2)
Variable set in multiple target state machines by one source state machine.
Variable set in one target state machine by one source state machine.
Variable set in one target state machine by one source state machine, then reset by target state machine.

Figure 8-1—Relationships among LRP-DT and LRP-DS state machines

Every Portal in a system has a unique value for the 5-tuple {pamAppId, pamMyChassisId, pamMyPortId, pamNeighborChassisId, pamNeighborPortId}.

Note that, as described in Clause 7 and Clause 8, the first Hello LRPDU from a neighbor could take a path through the application instance, while subsequent Hello LRPDU could be processed quickly by the Receive Hello state machine (8.2.5). An implementation shall ensure that Hello LRPDUs received from one LRP-DT instance are presented to pamReceiveHello in the order received.

8.2.2 Portal state machine variables

8.2.2.1 pamAppId

The value transmitted in, and expected to be received in, the appId field (9.4.2.1) of Hello LRPDUs.

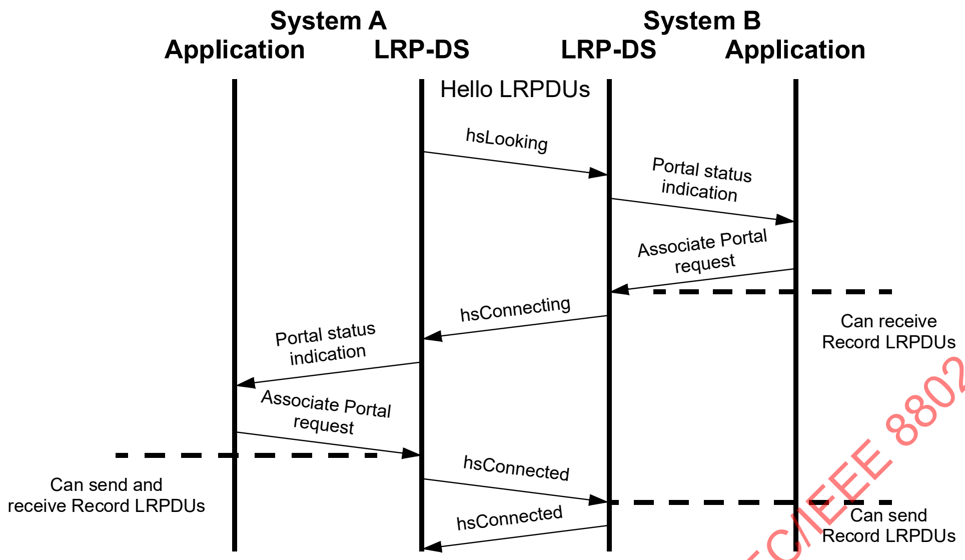


Figure 8-2—Typical three-way Hello handshake

8.2.2.2 pamMyChassisId

The value transmitted in the My Chassis ID TLV (9.4.2.6), and expected to be received in the Neighbor Chassis ID TLV (9.4.2.8), of Hello LRPDUs.

8.2.2.3 pamMyPortId

The value transmitted in the My Port ID TLV (9.4.2.7), and expected to be received in the Neighbor Port ID TLV (9.4.2.9), of Hello LRPDUs.

8.2.2.4 pamNeighborChassisId

The value transmitted in the Neighbor Chassis ID TLV (9.4.2.8), and expected to be received in the My Chassis ID TLV (9.4.2.6), of Hello LRPDUs.

8.2.2.5 pamNeighborPortId

The value transmitted in the Neighbor Port ID TLV (9.4.2.9), and expected to be received in the My Port ID TLV (9.4.2.7), of Hello LRPDUs.

8.2.2.6 pamNeighborPortalNumberValid

A Boolean value indicating whether a value for pamNeighborPortalNumber is valid (TRUE) or has not yet been received in a Hello LRPDU.

8.2.2.7 pamNeighborPortalNumber

The value received in the My Portal Number field (9.4.2.4) of the Hello LRPDU that established this Portal, and in the My Portal Number of all other LRPDUs (9.4.3, 9.4.4, 9.4.5) from that same neighbor Portal.

8.2.2.8 pamMyHelloStatus

An enumerated value to be transmitted in the Hello status field (9.4.2.2) of any Hello LRPDU. pamMyHelloStatus can take the following values:

hsLooking	This Portal has not yet received a successful Associate Portal request (10.2.4).
hsConnecting	This Portal has received a successful Associate Portal request (10.2.4), and a Hello LRPDU with the hsLooking status. The Portal is ready to receive all LRPDUs.
hsConnected	This Portal is up and ready to transfer LRP application data. The Portal is allowed to transmit all LRPDUs.

8.2.2.9 pamMyPortalNumber

A 32-bit portal number to be transmitted in the My Portal Number (9.4.2.4) of the Hello LRPDU. A system shall not have more than one Portal state machine with the same values of pamAppId, pamMyChassisId, and pamMyPortalNumber.

8.2.2.10 pamLocalOverflow

A Boolean value. Contains the last Boolean input (item b in 10.3.2.2.1) from the Database overflow request (10.3.2.2). A value of TRUE indicates that the partner applicant database has exceeded the capacity of the local registrar application instance. Transmitted in the database overflow bit [bit 8 (least significant)] in the Error status field (9.4.2.3) of Hello LRPDUs.

8.2.2.11 pamNeighborOverflow

A Boolean copied from the last-received database overflow bit [bit 8 (least significant)] in the Error status field (9.4.2.3) of the last-received Hello LRPDU.

8.2.2.12 pamNeighborAcknowledged

A Boolean, equal to the AND of all of the actAcknowledged variables (8.3.2.8) for all of the Applicant state machines (records) on this Portal.

8.2.2.13 pamNeedHello

A Boolean indicating that a Hello LRPDU needs to be sent.

8.2.2.14 pamResetReceiveHello

A Boolean indicating that the Receive Hello state machine needs to be reset.

8.2.2.15 pamHelloTime

Value to put in the Hello Time field (9.4.2.5) of a transmitted Hello LRPDU. See 9.4.2.5 for maximum and minimum values for this variable. A value of zero indicates that the Send Hello state machines (8.2.4) is to transmit one last Hello LRPDU (with a zero Hello Time field) and then cease transmitting.

8.2.2.16 pamMyAppInfo

Value to put in the Application Information TLV (9.4.2.10) of a transmitted Hello LRPDU.

8.2.2.17 pamHelloSendTimer

An integer valued timer that decrements once every second if and only if its value is non-zero.

8.2.2.18 pamHelloReceiveTimer

An integer valued timer that decrements once every second if and only if its value is non-zero.

8.2.2.19 pamHelloReceived

A Boolean indicating that a Hello LRPDU (9.4.2) has been received.

8.2.2.20 pamAssociationYes

A Boolean indicating that an Associate Portal request (10.2.4) returned TRUE.

8.2.2.21 pamAssociationNo

A Boolean indicating that an Associate Portal request (10.2.4) returned FALSE.

8.2.2.22 pamLastReceivedStatus

Used by the Receive Hello state machine (8.2.5) to record the Hello status field (9.4.2.2) of a Hello LRPDU received from the neighbor Portal.

8.2.2.23 pamKill

Boolean. When TRUE, causes the Receive Hello state machine (8.2.5) to terminate operation of the Portal.

8.2.3 Portal Association maintenance routines

8.2.3.1 pamAssociatePortal

This routine is called when the Associate Portal request (10.2.4) is issued. If the association is accepted, then pamAssociatePortal sets pamAssociationYes (8.2.2.20), else it sets pamAssociationNo (8.2.2.21).

8.2.3.2 pamResetNeighbor

This routine resets the neighbor information, if necessary. If the Portal is attached to an ECP LRP-DT instance, and if either the imPplExploreXmit or imPplExploreRecv flags in the imPplNeighborList (item h:4, item h:5 in 7.2.2.1) that created this Portal are TRUE, then pamResetNeighbor resets the imPplNbrChassisId and imPplNbrPortId items in the imPplNeighborList to null, thus enabling a subsequent exploratory Hello LRPDU (6.6) to create an association with another neighbor.

8.2.3.3 pamSendHello

Used by the Send Hello state machines (8.2.4) to transmit a Hello LRPDU (9.4.2) that it has constructed as follows:

a) Copy pamAppId (8.2.2.16) to the appId field (9.4.2.1).
b) Copy pamMyHelloStatus (8.2.2.8) to the Hello status (9.4.2.2).
c) Copy pamLocalOverflow (8.2.2.10) to the Error status (9.4.2.3).
d) Copy pamMyPortalNumber (8.2.2.9) to the My Portal Number (9.4.2.4).

e) Copy pamHelloTime (8.2.2.15) to the Hello Time (9.4.2.5).

f) Copy pamMyChassisId (8.2.2.2) to the My Chassis ID TLV (9.4.2.6).

g) Copy pamMyPortId (8.2.2.3) to the My Port ID TLV (9.4.2.7).

h) If pamNeighborChassisId (8.2.2.4) is not null, copy pamNeighborChassisId to the Neighbor Chassis ID TLV (9.4.2.8), else do not place a Neighbor Chassis ID TLV in the Hello LRPDU.

i) If pamNeighborPortId (8.2.2.5) is not null, copy pamNeighborPortId to the Neighbor Port ID TLV (9.4.2.9), else do not place a Neighbor Port ID TLV in the Hello LRPDU.

j) Copy pamMyAppInfo (8.2.2.16) to the Application Information TLV (9.4.2.10).

k) Set the TLV Length field (9.3.3) to the total size of all of the fields a through j.

l) Set the LRPDU type field (9.3.2) with typeHelloLRPDU (1).

8.2.3.4 pamReceiveHello

When instReceiveLRPDU (7.3.3.1) has successfully identified the Portal to which a received Hello LRPDU (9.4.2) belongs, it passes that Hello LRPDU to pamReceiveHello, which performs the following actions:

a) The value in the My Portal Number field replaces the value of pamNeighborPortalNumber (8.2.2.7) and pamNeighborPortalNumberValid is set TRUE.

NOTE—instReceiveLRPDU (7.3.3.1) selects a Portal to process a received Hello LRPDU based on the Neighbor Chassis ID TLV and Neighbor Port ID TLV in the Hello LRPDU. The My Portal Number field contained in the Hello LRPDU is used to identify which Portal is to process all other LRPDU's received subsequent to the Hello LRPDU. Thus, if the Portal Number changes, perhaps because the transmitting system has been reset, the change does not disrupt the operation of the receiving system.

b) pamReceiveHello then examines the Error status field (9.4.2.3) of the Hello LRPDU. If pamMyHelloStatus (8.2.2.8) has the value hsConnecting or hsConnected, and if pamNeighborOverflow (8.2.2.11) is not equal to the value of the database overflow bit [bit 8 (least significant) of the Error status field], then pamReceiveHello stores the new value of the database overflow into pamNeighborOverflow, and issues a Portal status indication (10.2.6) indicating the new state of pamNeighborOverflow.

c) Finally, pamReceiveHello presents the Hello LRPDU to the Receive Hello state machine (8.2.5) by setting the pamHelloReceived flag (8.2.2.19).

8.2.4 Send Hello state machines

The Send Hello state machines is defined in Figure 8-3.

8.2.5 Receive Hello state machine

The Receive Hello state machine is defined in Figure 8-4. Some explanatory notes about the state machine follow:

a) The RH_KILL state is an independent state machine that executes when pamKill is set.

b) On an ECP LRP-DT instance, the rules in instReceiveLRPDU (7.3.3.1) can result in offering a new Hello LRPDU to the application instance via the First Hello indication even though a Portal already is in operation on that ECP LRP-DT instance. But, in general, multiple Portals cannot be supported by ECP. It is up to the application instance to use the primitives supplied in Clause 10 to decide which neighbor to associate with and which to decline.

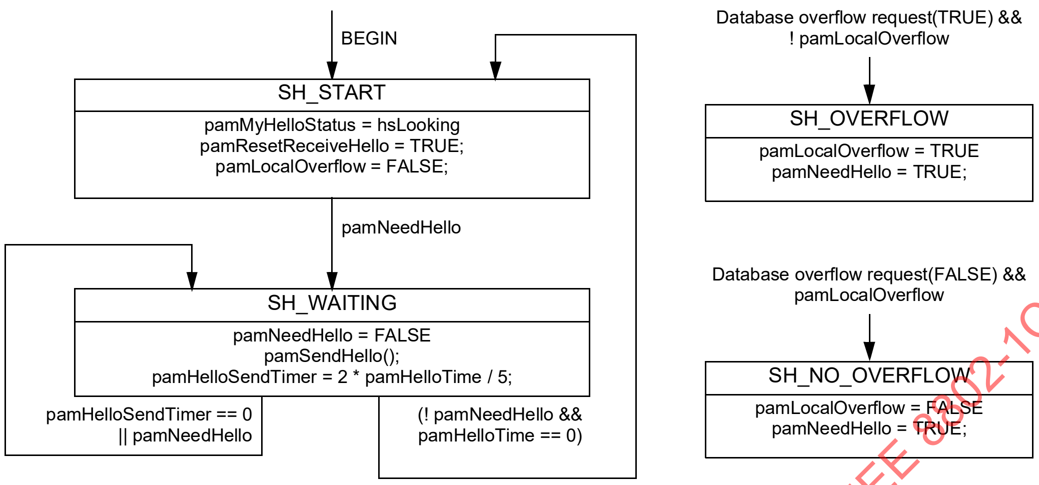

BEGIN
SH_START
pamMyHelloStatus = hsLooking
pamResetReceiveHello = TRUE;
pamLocalOverflow = FALSE;
pamNeedHello
SH_WAITING
pamNeedHello = FALSE
pamSendHello();
pamHelloSendTimer = 2 * pamHelloTime / 5;
pamHelloSendTimer == 0 || pamNeedHello
(! pamNeedHello && pamHelloTime == 0)
Database overflow request(TRUE) && ! pamLocalOverflow
SH_OVERFLOW
pamLocalOverflow = TRUE
pamNeedHello = TRUE;
Database overflow request(FALSE) && pamLocalOverflow
SH_NO_OVERFLOW
pamLocalOverflow = FALSE
pamNeedHello = TRUE;

Figure 8-3—Send Hello state machines

8.3 Applicant

8.3.1 Applicant overview

There is an instance of the Applicant state machine (8.3.4) for each record in the applicant database maintained by a Portal, and an additional set of Applicant state machines for each Portal. The state machines are created and destroyed as records are created and destroyed by the application instance. All of these state machines reference the pamMyHelloStatus variable (8.2.2.8) to ensure that LRPDUs are transmitted only when the Portal is properly associated.

The Portal interface offered by LRP-DS to the LRP application layer for operation of Portals are defined in 10.3.

8.3.2 Per-record applicant database variables

8.3.2.1 actRecordNumber

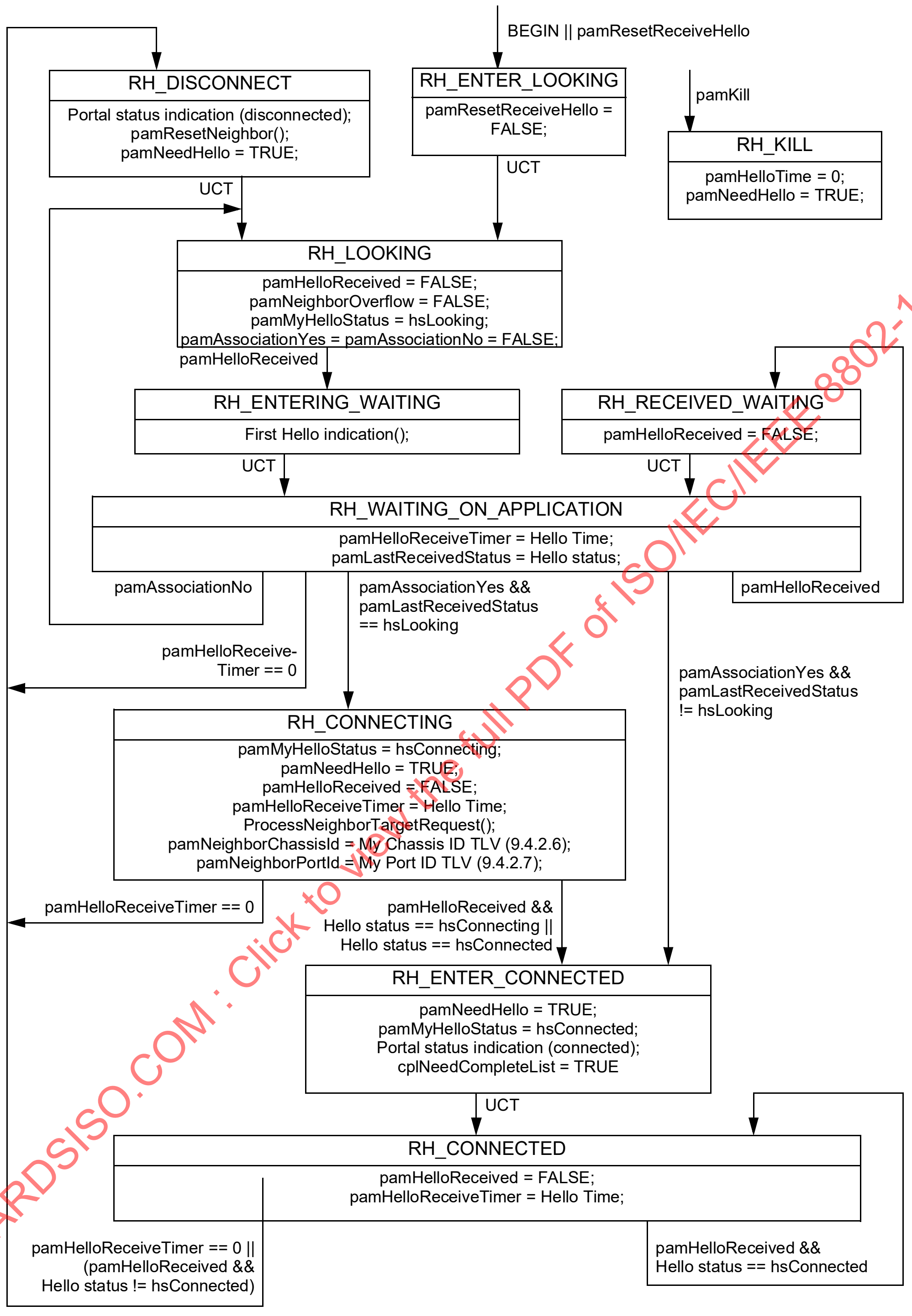
An integer in the range 0 through 4 294 967 295, inclusive, that uniquely identifies a single record in an applicant database and the corresponding record in the neighbor Portal's registrar database. There is no relationship between a record in an applicant database and a record with the same actRecordNumber in the same Portal's registrar database, if any.

8.3.2.2 actSequenceNumber

An integer in the range 0 through 4 294 967 295, inclusive, that identifies the version of the record. It is incremented each time the record changes.

8.3.2.3 actChecksum

A 16-bit checksum, computed according to 9.4.6, over the record. This variable can contain the value 0 only if there is no data in the record (actDataLength, 8.3.2.4, is 0).


BEGIN || pamResetReceiveHello
RH_DISCONNECT
Portal status indication (disconnected);
pamResetNeighbor();
pamNeedHello = TRUE;
RH_ENTER_LOOKING
pamResetReceiveHello = FALSE;
pamKill
RH_KILL
pamHelloTime = 0;
pamNeedHello = TRUE;
UCT
UCT
RH_LOOKING
pamHelloReceived = FALSE;
pamNeighborOverflow = FALSE;
pamMyHelloStatus = hsLooking;
pamAssociationYes = pamAssociationNo = FALSE;
pamHelloReceived
RH_ENTERING_WAITING
First Hello indication();
RH_RECEIVED_WAITING
pamHelloReceived = FALSE;
UCT
UCT
RH_WAITING_ON_APPLICATION
pamHelloReceiveTimer = Hello Time;
pamLastReceivedStatus = Hello status;
pamAssociationNo
pamAssociationYes &&
pamLastReceivedStatus
== hsLooking
pamHelloReceived
pamHelloReceive-
Timer == 0
pamAssociationYes &&
pamLastReceivedStatus
!= hsLooking
RH_CONNECTING
pamMyHelloStatus = hsConnecting;
pamNeedHello = TRUE;
pamHelloReceived = FALSE;
pamHelloReceiveTimer = Hello Time;
ProcessNeighborTargetRequest();
pamNeighborChassisId = My Chassis ID TLV (9.4.2.6);
pamNeighborPortId = My Port ID TLV (9.4.2.7);
pamHelloReceiveTimer == 0
pamHelloReceived &&
Hello status == hsConnecting ||
Hello status == hsConnected
RH_ENTER_CONNECTED
pamNeedHello = TRUE;
pamMyHelloStatus = hsConnected;
Portal status indication (connected);
cplNeedCompleteList = TRUE
UCT
RH_CONNECTED
pamHelloReceived = FALSE;
pamHelloReceiveTimer = Hello Time;
pamHelloReceiveTimer == 0 ||
(pamHelloReceived &&
Hello status != hsConnected)
pamHelloReceived &&
Hello status == hsConnected

NOTE—In this figure, “Hello status” is the value of the Hello status (9.4.2.2) of the received Hello LRPDU

Figure 8-4—Receive Hello state machine

8.3.2.4 actDataLength

The number of octets, 0 through 65 535, inclusive, in the actData variable (8.3.2.5). If 0, then actData is empty, and actChecksum (8.3.2.3) contains 0.

8.3.2.5 actData

An octet string of length actDataLength (8.3.2.4), containing the data in the record, if any.

8.3.2.6 actRecordChanged

A Boolean used by actWriteRecord (8.3.3.1) and actReceivePartialList (8.3.3.3) to indicate to the Applicant state machine that a record has changed and needs to be transmitted in a Record LRPDU (9.4.3).

8.3.2.7 actResendRecord

A Boolean used by actReceivePartialList (8.3.3.3) to indicate to the Applicant state machine (8.3.4) that a record needs to be retransmitted in a Record LRPDU (9.4.3).

8.3.2.8 actAcknowledged

A Boolean that is set FALSE when a record is transmitted in a Record LRPDU (9.4.3), and set TRUE when the record is acknowledged by a Partial List LRPDU (9.4.4) or a Complete List LRPDU (9.4.5).

8.3.3 Applicant routines

8.3.3.1 actWriteRecord

This routine is called whenever the application instance issues the primitive Write record request (10.3.1.1). actWriteRecord performs the following actions:

a) It checks to see if any existing Applicant state machine's actRecordNumber variable (8.3.2.1) matches the record number in the Write record request (item b in 10.3.1.1.1). If not, it creates a new Applicant state machine using Write record request to supply a value for actRecordNumber in the new state machine.
b) It computes new values for actChecksum (8.3.2.3), actDataLength (8.3.2.4), and actData (8.3.2.5) for the selected Applicant state machine from the data given in the Write record request (item c in 10.3.1.1.1).
c) It sets the selected Applicant state machine's actRecordChanged variable (8.3.2.6) to TRUE.

8.3.3.2 actSendRecord

This routine triggers the transmission of a Record LRPDU (9.4.3) containing the information for this Applicant state machine's record. actSendRecord creates a record for the Record LRPDU according to Table 9-7, copying the actRecordNumber (8.3.2.1), actSequenceNumber (8.3.2.2), actChecksum (8.3.2.3), actDataLength (8.3.2.4), and actData (8.3.2.5) to the record's Record number, Sequence Number, Checksum, Data length, and Application data fields, respectively. This record can be placed into its own Record LRPDU (9.4.3) and transmitted over the LRP-DT instance (7.3), or it can be combined with other records from other Write record requests into a larger Record LRPDU and then transmitted.

8.3.3.3 actReceivePartialList

Whenever a Partial List LRPDU (9.4.4) is received on a Portal, the actReceivePartialList routine is called once for each record header (Table 9-9) received, or when called by actReceiveCompleteList (8.3.3.4), actReceivePartialList performs the following actions:

a) If the pamMyHelloStatus variable (8.2.2.8) is in the hsLooking state, the Partial List LRPDU is discarded without examination.

b) If the Record number field does not match any of the actRecordNumber variables (8.3.2.1) of the Applicant state machines on this Portal, and the received Checksum is zero, then the received record header is ignored. (The neighbor Portal is acknowledging a deleted record.)

c) If the Record number field does not match any of the actRecordNumber variables (8.3.2.1) of the Applicant state machines on this Portal, and the received Checksum is non-zero, then a new Applicant state machine is created with its variables set according to Table 8-1, and processing terminates. (The neighbor Portal somehow failed to receive notification of a deleted record.)

Table 8-1—New Applicant state machine from unexpected Partial List LRPDU

Set variable (8.3.2)	from record header	or from constant
actRecordNumber (8.3.2.1)	Record number	
actSequenceNumber (8.3.2.2)	Sequence Number + 1	
actChecksum (8.3.2.3)		0
actDataLength (8.3.2.4)		0
actData (8.3.2.5)		none
actRecordChanged (8.3.2.6)		TRUE

d) Next (which will not happen unless the Record number field does match a actRecordNumber variable), the Sequence Number field of the received header is compared to the actSequenceNumber variable (8.3.2.2) of the matching Applicant state machine. If the received number is smaller than the actSequenceNumber, actReceivePartialList sets the actResendRecord variable (8.3.2.7), and processing terminates. (The record header acknowledges an old version of the current record.)

e) Next (which will not happen unless the Record number field does match a actRecordNumber variable and the received Sequence Number is >= actSequenceNumber), if the received Sequence Number field is greater than the actSequenceNumber variable, then actSequenceNumber is set to the higher of the two numbers, actRecordChanged is set TRUE, and processing terminates. (The local and neighbor state machines have a different opinion of the sequence number. We will make sure that the registrar state machine sees a new sequence number.)

f) Next (the Record number and Sequence Number fields match) the Checksum field is compared to the value in actChecksum (8.3.2.3). If the values match, then the actAcknowledged is set TRUE. If the value in actChecksum is zero, then a deleted record has been acknowledged, and actTerminateStateMachine (8.3.3.5) is called to destroy this record’s Applicant state machine. Otherwise (actChecksum is not zero), processing terminates. (The record has been properly acknowledged.)

g) Otherwise (everything but the checksum matches), actRecordChanged is set TRUE. (The record has been improperly acknowledged and must be resent.)

8.3.3.4 actReceiveCompleteList

This routine is called once to process an entire Complete List LRPDU (9.4.5) received for this Portal. actReceiveCompleteList performs the following actions, calling actReceivePartialList (8.3.3.3) as necessary to process the record headers in the Complete List LRPDU.

a) If the pamMyHelloStatus variable (8.2.2.8) is in the hsLooking state, the Complete List LRPDU is discarded without examination.
b) For each possible record number between the First record number and Last record number fields (Table 9-9), inclusive, for which there is no corresponding record header, actReceiveCompleteList calls actReceivePartialList (8.3.3.3) with a simulated record header with that record number, a zero sequence number, and a zero checksum.

NOTE—The preceding paragraph defines what the outcome of the processing is to be. An actual implementation can presumably find a better algorithm than to execute 4 billion subroutine calls.

c) Any record header in the Complete List LRPDU whose Record number field is outside the range of First record number and Last record number is ignored.
d) All other record headers are passed to actReceivePartialList for processing.

8.3.3.5 actTerminateStateMachine

This routine is called by actReceivePartialList (8.3.3.3) to terminate the Applicant state machine (8.3.4) for one record number (actRecordNumber, 8.3.2.1), destroying all of its variables (8.3.2).

8.3.4 Applicant state machine

The Applicant state machine is defined in Figure 8-5.

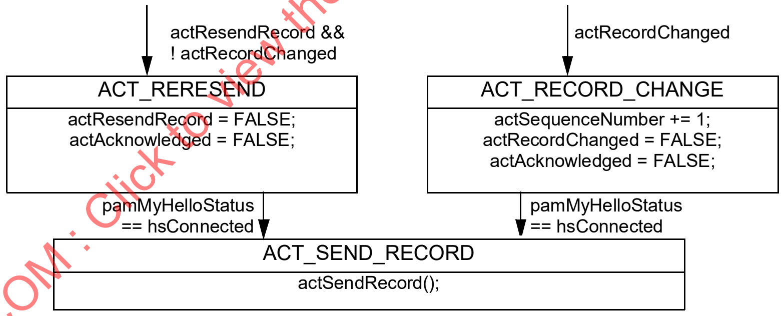


Figure 8-5—Applicant state machine

8.4 Registrar

8.4.1 Registrar overview

There is a single instance of the Partial list state machine (8.4.5). This state machine has its own set of variables (8.4.2), and also an array of variables (8.4.3), one set for each record in the registrar database maintained by the Portal. The variables are created and destroyed as records are created and destroyed by the neighbor application instance via the transmission of Record LRPDUs (9.4.3), and by the application instance, via the Delete record request primitive (10.3.2.1). The Partial list state machine is responsible for transmitting acknowledging Partial List LRPDUs (9.4.4) in response to the received Record LRPDUs. There is also a single Complete list state machine (8.4.8) that is used to transmit Complete List LRPDUs (9.4.5), both on demand and periodically.

All of these state machines reference the pamMyHelloStatus variable (8.2.2.8) to ensure that LRPDUs are transmitted only when the Portal is properly associated.

8.4.2 Per-Portal registrar variables

8.4.2.1 ptlNeedPartialList

A Boolean that can be set to TRUE to initiate the transmission of one or more Partial List LRPDUs when a Record LRPDU (9.4.3) is received.

8.4.3 Per-record registrar variables

8.4.3.1 regRecordNumber

An integer in the range 0 through 4 294 967 295, inclusive, that uniquely identifies a single record in a registrar database, and the corresponding record in the neighbor Portal's applicant database. There is no relationship between a record in a registrar database and a record with the same regRecordNumber in the same Portal's applicant database, if any.

8.4.3.2 regSequenceNumber

An integer in the range 0 through 4 294 967 295, inclusive, that identifies the version of the record. It is copied from the received Record LRPDU (9.4.3).

NOTE—If regSequenceNumber wraps around back 0, then the applicant-side LRP-DS will be unable to update the record again. (The registrar will see 0 as smaller than 4 294 967 295, re-acknowledge the higher number and not change the registrar database.)

8.4.3.3 regChecksum

A 16-bit checksum, computed according to 9.4.6, over the record. It is copied from the received Record LRPDU (9.4.3).

8.4.3.4 regSendAck

A Boolean indicating that this header for this record needs to be transmitted in a Partial List LRPDU (9.4.4).

8.4.4 Registrar routines

8.4.4.1 regReceiveWriteRecord

This routine is called whenever Record LRPDU (9.4.3) is received for this Portal. It is called once for each record (Table 9-7) contained in the Record LRPDU. regReceiveWriteRecord performs the following actions:

a) If the pamMyHelloStatus variable (8.2.2.8) is in the hsLooking state, the Record LRPDU is discarded without examination.

b) Otherwise (pamMyHelloStatus is not hsLooking), the receiving system should compute a checksum on the record and compare it to the receive Checksum field. If a checksum mismatch is detected, or if only one of the Checksum field and the Data length field of the record are zero, then the record shall be discarded, one error counted in RecordErrors (11.5.9), and the registrar database left unchanged. No further processing of the record is performed.

NOTE 1—Because the LRPDUs are carried over a reliable transport protocol as a byte stream, there is no point in triggering a retransmission for a bad record Checksum field. The bad record is not acknowledged in a Partial List LRPDU; the Applicant will retry the record transmission after the next Complete List LRPDU. An alarm can be sent to a network management facility when RecordErrors is incremented, but such alarms are outside the scope of this standard.

c) Otherwise (the record in the Record LRPDU is valid), regReceiveWriteRecord finds state machine whose regRecordNumber variable (8.4.3.1) matches the Record number field in the received Record LRPDU. If none is found, a new Partial list state machine is created for that record, using the information in the Record LRPDU.

d) regReceiveWriteRecord then compares the Sequence Number field (Table 9-7) of the received record to the regSequenceNumber variable (8.4.3.2). If the received Sequence Number is higher than regSequenceNumber, then regReceiveWriteRecord copies the Sequence Number and Checksum fields to regSequenceNumber and regChecksum, and generates a Record written indication (10.3.2.3) to notify the application instance of the received data (or record deletion).

NOTE 2—If the received Sequence Number is not higher than regSequenceNumber, then item e causes the old, recorded information for this record to be transmitted in a Partial List LRPDU.

e) Next, regReceiveWriteRecord sets regSendAck (8.4.3.4) and ptlNeedPartialList (8.4.2.1) to TRUE.

8.4.4.2 regSendPartialList

This routine creates an Partial List LRPDU, or multiple Partial List LRPDUs, if the number of records to be transmitted will not fit into a single Partial List LRPDU, and queues them for transmission. The Partial List LRPDUs contain one record header (Table 9-9) for each record in the registrar database whose regSendAck variable (8.4.3.4) is TRUE. The regSendAck variables are set to FALSE. After setting regSendAck to FALSE, if the regChecksum variable for that record (8.4.3.3) is zero, the record is deleted from the database.

8.4.4.3 regDeleteRecord

This routine is called whenever the Delete record request primitive (10.3.2.1) is issued by the application instance. It deletes the state machine whose regRecordNumber variable (8.4.3.1) matches the record number specified in the Delete record request primitive, and sets cplNeedCompleteList (8.4.6.1) to TRUE to trigger the transmission of a Complete List LRPDU (9.4.5).

8.4.5 Partial list state machine

The Partial list state machine is defined in Figure 8-6.

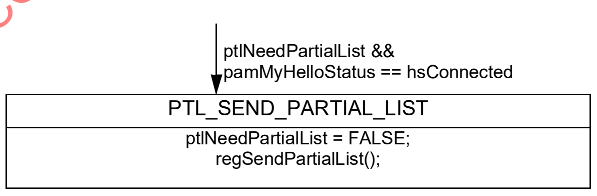


Figure 8-6—Partial list state machine

8.4.6 Complete list variables

8.4.6.1 cplNeedCompleteList

A Boolean that can be set to TRUE to initiate the transmission of one or more Complete List LRPDUs as soon as the Portal association is complete.

8.4.6.2 cplCompleteListTimer

An integer valued timer that decrements once every second, if and only if its value is non-zero.

8.4.6.3 cplCompleteListTimerReset

An integer used by cplResetCplTimer (8.4.7.1) to reset the value of cplCompleteListTimer (8.4.6.2). cplCompleteListTimerReset is initialized from the Local Target Port request (item c:3 in 10.2.2.1) that created the target port.

8.4.7 Complete list routines

8.4.7.1 cplResetCplTimer

Sets cplCompleteListTimer (8.4.6.2) to the value cplCompleteListTimerReset + *x* * cplCompleteListTimerReset, where "*x*" is a pseudo-random value $0 \leq x < 1$.

8.4.7.2 cplSendCompleteList

This routine creates an Complete List LRPDU, or multiple Complete List LRPDUs, if the number of records to be transmitted will not fit into a single Complete List LRPDU. The Complete List LRPDUs contain one record header (Table 9-9) for each record in the registrar database whose regChecksum variable (8.4.3.3) is non-zero. The set of Complete List LRPDUs transmitted span the entire 32-bit space of the record number space, as described in 9.4.5.

8.4.8 Complete list state machine

The Complete list state machine is illustrated in Figure 8-7.

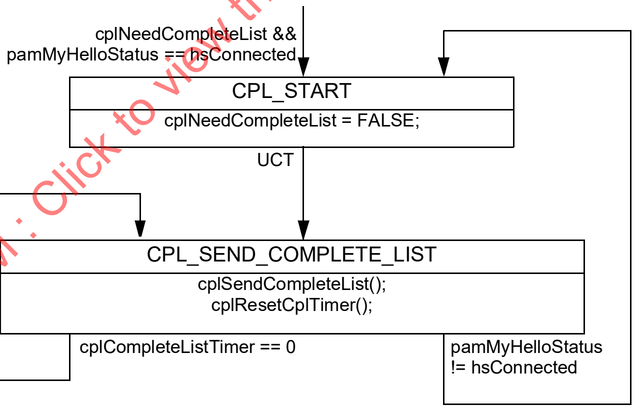

cplNeedCompleteList && pamMyHelloStatus == hsConnected
CPL_START
cplNeedCompleteList = FALSE;
UCT
CPL_SEND_COMPLETE_LIST
cplSendCompleteList();
cplResetCplTimer();
cplCompleteListTimer == 0
pamMyHelloStatus != hsConnected

Figure 8-7—Complete list state machine

9. Format and encoding of LRP Data Units

9.1 Introduction

This clause specifies how Link-local Registration Protocol Data Units (LRPDUs) are encoded for transmission over the LRP Data Transport (LRP-DT) mechanisms (6.9):

a) Subclause 9.2 describes the format of an AppId, which is used in a number of contexts besides LRPDUs.
b) Subclause 9.3 describes the common format and encoding of all LRPDUs.
c) Subclause 9.4 describes the format and encoding of each individual LRPDU message.
d) Subclause 9.5 describes how the LRPDUs are sent and received over LRP-DT.

9.2 AppId

An AppId is used in several subclauses in this document (9.4.2.1, 10.2.2.1, C.2.1.6, C.2.2.6) to identify a specific LRP application. (See B.11.) Its format is shown in Table 9-1. The OUI or CID field identifies the organization that is responsible for defining the content of the Application Sub-ID field. The value of the OUI or CID field is an OUI or a CID.[13] It is the responsibility of the OUI or CID owner to manage the use of the Application Sub-ID.

Table 9-1—AppId format

Field	Length (octets)	Offset (octets)
OUI or CID	3	0
Application Sub-ID	1	3
		4

9.3 LRP database synchronization protocol

Any number of LRPDUs can be strung together successively in a single LRP-DT Edge Control Protocol Data Unit (ECPDU), up to the maximum size of the layer 2 frame. A single LRPDU cannot be split across multiple ECPDUs. When using TCP, an LRPDU's size is limited by its 16-bit length field.

9.3.1 LRPDU format

Every LRPDU except the Stop LRPDU is encoded in a Type/Length/Value (TLV) format. The TLV format shown in Table 9-2 supports data fields of up to 65 535 octets.

The Stop LRPDU (9.4.1), and only that TLV type, has an LRPDU type field, but no TLV Length field and no data.

9.3.2 LRPDU type field

Table 9-3 shows the meaning of the values in the LRPDU type field of an LRPDU.

[13]See IEEE Std 802. An OUI or CID is assigned to an organization by the IEEE registration authority. OUIs and CIDs are drawn from different parts of a 24-bit space, but whether the value used in an AppId is an OUI or a CID does not affect the operation of the protocols specified in this document.

Table 9-2—LRPDU TLV format

Field	Length (octets)	Offset (octets)
Type (9.3.2)	1	0
Length (9.3.3)[a]	2	1
Data[a]	0-65 535	3

[a]Not present in a Stop LRPDU (Type field = typeStopLRPDU).

Table 9-3—LRPDU type field values for LRPDUs

Value	Name	Reference
0	typeStopLRPDU	9.4.1
1	typeHelloLRPDU	9.4.2
2	typeRecordLRPDU	9.4.3
3	typePartialListLRPDU	9.4.4
4	typeCompleteListLRPDU	9.4.5
5	typeMyChassisId	9.4.2.6
6	typeMyPortId	9.4.2.7
7	typeNeighborChassisId	9.4.2.8
8	typeNeighborPortId	9.4.2.9
9	typeAppInfo	9.4.2.10
10–255	Reserved	

9.3.3 TLV Length field

A two-octet integer, with the most-significant 8 bits in the first octet (offset 1), that contains the length of the data field. Thus, a TLV Length field of 0 indicates that no data field is present.

9.4 LRPDU formats

9.4.1 Stop LRPDU

The Stop LRPDU contains only a LRPDU type field (9.3.2), with the value typeStopLRPDU (0). Thus, an LRPDU can be padded with any number of octets of 0, and each octet can be interpreted as a Stop LRPDU.

The Stop LRPDU carries no semantic content; its purpose is purely to provide padding that may be necessary to fill out an ECPDU to a minimum length.

9.4.2 Hello LRPDU

LRPDU type field (9.3.2) = typeHelloLRPDU. The data field of the Hello LRPDU contains information transmitted by a single instance of the Send Hello state machines (8.2.4). The format of a Hello LRPDU is shown in Table 9-4.

Table 9-4—Hello LRPDU format

Field		Length (octets)	Offset (octets)	
Type (9.3.2)		1	0	
Length (9.3.3)		2	1	
appId field (9.4.2.1)		4	3	
Hello status (9.4.2.2)	Error status (9.4.2.3)	1	7	
My Portal Number (9.4.2.4)		4	8	fixed length
Hello Time (9.4.2.5)		2	12	
Two or four TLVs, My Chassis ID TLV (9.4.2.6), My Port ID TLV (9.4.2.7), Neighbor Chassis ID TLV (9.4.2.8), and Neighbor Port ID TLV (9.4.2.9), in any order.		variable	14	TLVs
0 or 1 Application Information TLV (9.4.2.10)		variable	variable	
			variable	

A Hello LRPDU contains a few fixed-size fields, followed by a series of TLVs within its data field. That is, the tenth octet following the TLV Length field is another LRPDU type field. The length field of the Hello LRPDU (octets 1 and 2, following the type field) contains the length of all of the fixed length fields plus all the TLVs in the data field. In Exploratory Hello LRPDUs (6.6), the first two TLVs in the Hello LRPDU are one each of the My Chassis ID TLV (9.4.2.6) and My Port ID TLVs (9.4.2.7). In all other Hello LRPDUs, the first four TLVs in the Hello LRPDU are one each of the My Chassis ID TLV (9.4.2.6), My Port ID TLV (9.4.2.7), Neighbor Chassis ID TLV (9.4.2.8), and Neighbor Port ID TLV (9.4.2.9), in any order. Following these comes either 0 or 1 Application Information TLV (9.4.2.10).

9.4.2.1 appId field

The appId field identifies the LRP application associated with the transmitting Portal. Its format is defined in 9.2.

9.4.2.2 Hello status

A 4-bit enumerated field, in the most-significant bits of the octet, containing one of the following values (see 8.2.2.8 for the meanings of these values):

0)	**hsLooking**
1)	**hsConnecting**
2)	**hsConnected**
3–15)	reserved

9.4.2.3 Error status

Four bits of information, in the least-significant bits of the octet, as described in Table 9-5.

Table 9-5—Error status bits

Bit	Transmits contents of	References
5	reserved	
6	reserved	
7	reserved	
8 (least significant)	database overflow	8.2.2.10, 8.2.2.11

The bits marked "reserved" shall be transmitted as 0, and shall not be examined on receipt.

9.4.2.4 My Portal Number

The My Portal Number field contains a four-octet number identifying the transmitting Portal (8) that is unique over all of the Portals sharing this same LRP-DT instance (7.3). See the descriptions of the Record LRPDU, Partial List LRPDU, and Complete List LRPDU (9.4.3, 9.4.4, 9.4.5) for its use in those LRPDUs.

NOTE 1—This field is four octets in length because a Proxy system can proxy for any number of Controlled systems, each of which can have a large number of target ports, and it can make a TCP connection to another, similar Proxy system. Each Proxy system requires one value of My Portal Number for each of the target ports for which it is proxying.

NOTE 2—There is no specified relationship between the My Portal Number field, the "PortalId" used in Clause 10, and the lrpPortalNumber object in the LRP MIB (13.5.2).

9.4.2.5 Hello Time

The Hello Time consists of two octets representing the time to live for the Hello LRPDU. The first octet of the Hello Time data field is the most-significant octet of a 16-bit number of seconds (0 or 30 to 65 535) that the Hello LRPDU is valid. This field shall not be transmitted with a value of between 1 and 29, inclusive.

9.4.2.6 My Chassis ID TLV

LRPDU type field (9.3.2) = typeMyChassisId. The data field of the My Chassis ID TLV is in the same format as the Chassis ID TLV in 8.5.2 of IEEE Std 802.1AB-2016. It contains the value of the pamMyChassisId variable (8.2.2.2) of the transmitting Portal.

9.4.2.7 My Port ID TLV

LRPDU type field (9.3.2) = typeMyPortId. The data field of the My Port ID TLV is in the same format as the Port ID TLV in 8.5.3 of IEEE Std 802.1AB-2016. It contains the value of the pamMyPortId variable (8.2.2.3) of the transmitting Portal.

9.4.2.8 Neighbor Chassis ID TLV

LRPDU type field (9.3.2) = typeNeighborChassisId. The data field of the Neighbor Chassis ID TLV is in the same format as the Chassis ID TLV in 8.5.2 of IEEE Std 802.1AB-2016. It contains the value of the pamNeighborChassisId variable (8.2.2.4) of the transmitting Portal.

9.4.2.9 Neighbor Port ID TLV

LRPDU type field (9.3.2) = typeNeighborPortId. The data field of the Neighbor Port ID TLV is in the same format as the Port ID TLV in 8.5.3 of IEEE Std 802.1AB-2016. It contains the value of the pamNeighborPortId variable (8.2.2.5) of the transmitting Portal.

9.4.2.10 Application Information TLV

LRPDU type field (9.3.2) = typeAppInfo. The data field of the Application Information TLV contains information supplied by the Local Target Port request (10.2.2). The data is presented to the application instance at the other end through the Portal status indication (10.2.6) at the other end of the LRP-DT connection. The data is opaque, and not interpreted, by LRP. The Application Information TLV is not present in the Hello LRPDU if not given in the Local Target Port request (10.2.2) primitive.

9.4.3 Record LRPDU

LRPDU type field (9.3.2) = typeRecordLRPDU. The format of the Record LRPDU is shown in Table 9-6.

Table 9-6—Record LRPDU format

Field	Length (octets)	Offset (octets)
Type (9.3.2)	1	0
Length (9.3.3)	2	1
My Portal Number (9.4.2.4)	4	3
Zero or more records	variable	7
		variable

The first four octets of a Record LRPDU data field is the My Portal Number field (9.4.2.4) that encodes the chassis ID, port ID, and AppId of the transmitting Portal. It is the same value last transmitted in a Hello LRPDU (9.4.2) for this same Portal. Following this are zero or more records. The format of each record is shown in Table 9-7. (See 9.4.6 for the computation of the checksum.)

Table 9-7—Format of one record within a Record LRPDU

Field	Length (octets)	Offset (octets)
Record number	4	0
Sequence Number	4	4
Checksum	2	8
Data length	2	10
Application data	0 to 65 520	12
		variable

9.4.4 Partial List LRPDU

LRPDU type field (9.3.2) = typePartialListLRPDU. The format of the Partial List LRPDU is shown in Table 9-8.

Table 9-8—Partial List LRPDU format

Field	Length (octets)	Offset (octets)
Type (9.3.2)	1	0
Length (9.3.3)	2	1
My Portal Number (9.4.2.4)	4	3
r record headers	10*r*	7
		7 + 10*r*

The first four octets of a Partial List LRPDU data field is a My Portal Number field (9.4.2.4) that encodes the chassis ID, port ID, and AppId of the transmitting Portal. It is the same value last transmitted in a Hello LRPDU (9.4.2) for this same Portal. Following this, there are zero or more record headers. The format of each record header is shown in Table 9-9; the record header of the Partial List LRPDU is identical to the record header of the Complete List LRPDU. (See 9.4.6 for the computation of the checksum.)

Table 9-9—Format of one record header

Field	Length (octets)	Offset (octets)
Record number	4	0
Sequence Number	4	4
Checksum	2	8
		10

9.4.5 Complete List LRPDU

LRPDU type field (9.3.2) = typeCompleteListLRPDU. The format of the Complete List LRPDU is shown in Table 9-10.

The first four octets of a Complete List LRPDU data field are a My Portal Number field (9.4.2.4) that encodes the chassis ID, port ID, and AppId of the transmitting Portal. It is the same value last transmitted in a Hello LRPDU (9.4.2) for this same Portal. Following this are two four-octet record numbers, which are the lowest and highest record number values that are encompassed by this Complete List LRPDU. Following this are zero or more record headers. The format of each record header is shown in Table 9-9.

The value in the Record number field of a record header transmitted in an Complete List LRPDU shall be greater than or equal to the value in the First record number field, and less than or equal to the value in the Last record number field, of that Complete List LRPDU.

Table 9-10—Complete List LRPDU format

Field	Length (octets)	Offset (octets)
Type (9.3.2)	1	0
Length (9.3.3)	2	1
My Portal Number (9.4.2.4)	4	3
First record number	4	7
Last record number	4	11
r record headers, $0 \leq r$	10*r*	15
		15 + 10*r*

The First record number and Last record number allow the complete list of records known to an Applicant state machine or a Partial list state machine to be split among more than one Complete List LRPDU. The pairs of first and last record numbers in all of the Complete List LRPDUs comprising a complete list of records span all possible record numbers from 0 through 4 294 967 295. For example, if record numbers 4, 5, 290, and 5122 comprise the whole list of records, they might be split into two Complete List LRPDUs as shown in Table 9-11.

Table 9-11—Complete List LRPDU example

LRPDU	First record number	Last record number	Record numbers included
1	0	10	4, 5
2	11	4 294 967 295	290, 5122

In this example, the first Complete List LRPDU says, "There are only two records with record numbers between 0 and 10, exclusive: 4 and 5."

9.4.6 Record checksum calculation

The Checksum field of a record (Table 9-9) is computed using the Fletcher algorithm [B1], specified in the following C code fragment. Note that the two-octet Data length field is included in the checksum as if it were the first two octets of the Application data field.

```
unsigned int (8) A = 0;
unsigned int (8) B = 0;
unsigned int (8) Checksum[2];   // record Checksum field
unsigned int (8) dataLength[2]; // record Data length field
unsigned int (8) data[65535];   // record Application data field
int length = int(dataLength[0] << 8) + dataLength[1];
int i;
unsigned int (8) A = dataLength[0];
unsigned int (8) B = A;
A = A + dataLength[1];
if (A < dataLength[1])
    A = A + 1;
B = B + A;
if (B < A)
    B = B + 1;
```

```
for (i = 0; i < length; i += 1) {
    A = A + data[i];     // 1's complement A = A + data[i]
    if (A < data[i])
        A = A + 1;
    B = B + A;           // 1's complement B = B + A
    if (B < A)
        B = B + 1;
}
Checksum[0] = A;
Checksum[1] = B;
```

When checking the checksum, the assignment of A and B to the Checksum is replaced by a comparison. See Nakassis [B20] and Sklower [B21] for more efficient computation algorithms than the preceding code fragment.

9.5 LRP data transport protocols

9.5.1 LRP-DT ECP protocol

See 6.9.1 for a list of incoming destination MAC addresses for frames carrying ECPDUs that are to be received by a system. The outgoing destination MAC address is taken from imPplNbrEcpInfo (item g:3 in 7.2.2.1).

The ECP Subtype (43.3.3.4 of IEEE Std 802.1Q-2018) shall have the value shown in Table 9-12, in order to indicate that the ECPDU carries one or more LRPDUs.

Table 9-12—ECP subtype field for LRP

ECP subtype
3 (decimal)

The first octet of the ULPDU field of an ECPDU (43.3.3.4 of IEEE Std 802.1Q-2018) shall be the first octet of an LRPDU. That is, an LRPDU cannot be split across more than one ECPDU.

If the transmission of one or more LRPDUs in an ECPDU would result in an ECPDU that is shorter than the minimum length required by the underlying medium, the transmitting LRP shall pad with one or more Stop LRPDUs until the frame meets the minimum length requirements.

9.5.2 LRP-DT TCP protocol

LRPDUs are serially encoded as user data.

The means for selecting the Port number are given in 6.9.2.3.

10. LRP-DS service interface

10.1 Introduction

Primitives for the interface between the LRP application layer and the LRP-DS layer are defined in this clause. They allow an application instance to create, destroy, and associate Portals (10.2), and provide for interaction between the application instance and its Portals (10.3). These primitives provide the events required to connect the state machines describing an LRP application to the LRP state machines in 7.2, 7.3, and Clause 8. The execution of these primitives is not directly visible outside a system, so there is no requirement for a system conformant to this standard to implement them explicitly. An actual implementation would be likely to include many parameters and operations that are not in this list of primitives. For example, the Local Target Port request primitive (10.2.2) and procedures only support creation and deletion, but not alteration, of a target port entry. This does not imply that an implementation cannot use more advanced operations.

To support these primitives, but not to specify how an implementation is to be constructed, the following assumptions are made:

a) The LRP-DS Portal maintains the complete and definitive applicant database. The Applicant database primitives (10.3.1) only specify changes to make to that database.

b) An application instance constructs and maintains the registrar database based on the operation of the registrar state machines (8.4), which use the Registrar database primitives (10.3.2) to indicate changes in the database to the application instance. The LRP-DS Portal maintains a complete list of record numbers, record sequence numbers, and record checksums. See also B.5.

The primitives in this clause use a "PortalId" (10.2.4, 10.2.6, 10.3.1.1, 10.3.2.1, 10.3.2.2, 10.3.2.3) to refer to a specific Portal. Because this service interface is not externally visible, the form of a PortalId is not specified by this standard. It could, for example, be a pointer to a data structure or an index into a table. There is no specified relationship among PortalId, My Portal Number (9.4.2.4), and lrpPortalNumber (in the LRP MIB, 13.5.2). The service interface is summarized in Table 10-1.

Table 10-1—LRP-DS service interface summary

Primitive	Defined	Direction	Purpose
Local Target Port request	10.2.2	Request	Provide local LRP-DT and Hello information
Neighbor Target Port request	10.2.2	Request	Provide neighbor's LRP-DT information
Associate Portal request	10.2.4	Request	Approve a Portal association after First Hello indication
First Hello indication	10.2.5	Indication	Query the application instance whether to allow Portal to associate
Portal status indication	10.2.6	Indication	Portal status has changed
Write record request	10.3.1.1	Request	Add/change record in applicant database and transfer to Registrar
Delete record request	10.3.2.1	Request	Delete record from Registrar database
Database overflow request	10.3.2.2	Request	Set state of Registrar database overflow
Record written indication	10.3.2.3	Indication	New/changed record has been received

10.2 Association primitives

10.2.1 Use of association primitives

The association primitives are used in the following order:

a) The application instance issues at least one Local Target Port request to initiate the creation of LRP-DT instances and Portals. This can cause Hello LRPDUs (9.4.2) to be received, and Portals can be created.

b) The application instance issues Neighbor Target Port requests to initiate the creation of LRP-DT instances and Portals.This can cause Hello LRPDUs (9.4.2) to be transmitted.

c) When a Hello LRPDU is received, the Portal state machine can issue a First Hello indication (10.2.5) to notify the application instance of an opportunity to create a Portal association. This indication invites the application instance to issue an Associate Portal request (10.2.4), with either a TRUE (create Portal Association) or FALSE (deny Portal Association). In the first case, the application instance can expect to receive Portal status indications (10.2.6) from the Applicant and/or Registrar state machines at any time. In the latter case, the Portal is not associated.

d) Portal status indications are made to the application instance to inform it of the progress of the status of the Portals, with one of the codes listed in item b in 10.2.6.1:

    1) The code "connected" indicates that the exchange of Hello LRPDU has completed successfully, and that the Portal is fully operational.

    2) The code "disconnected" indicates that the exchange of Hello LRPDUs has timed out or otherwise failed. However, the Portal still continues to operate, and another First Hello indication may be given in the future.

e) Once the Portal has been associated, the application instance can send request primitives to the Applicant and/or Registrar state machines at any time.

10.2.2 Local Target Port request

Local Target Port request is used to create or delete local target ports, and thus to initiate or terminate the process of creating LRP-DT instances and Portals for the issuing application instance. A Local Target Port request creates or deletes an entry in the imTargetPortList (7.2.2.1). Multiple Portals can be created from a single Local Target Port request, as entries in the imPplNeighborList (item g in 7.2.2.1) are added to the imTargetPortList entry with the Local Target Port request primitive (10.2.2). See 7.2.3.1 for a full description of the actions taken when a Local Target Port request is issued.

A system shall not accept a Local Target Port request that specifies a Chassis ID and Port ID that match those in another Local Target Port request.

10.2.2.1 Inputs

a) Request identification parameters:

    1) AppId: Identifies the requesting LRP application. See 9.2.

b) LLDP parameters:

NOTE 1—If the IEEE 802.1AB Link Layer Discovery Protocol (LLDP) is used, these are the parameters transmitted by the Native or Controlled system on the selected target port:

    1) The Chassis ID of the Native or Controlled system's local target port (8.5.2 of IEEE Std 802.1AB-2016).

    2) The Port ID of the Native or Controlled system's local target port (8.5.3 of IEEE Std 802.1AB-2016).

3) A Boolean indicating whether an LRP-DT instance using the LRP-DT ECP mechanism (6.9.1) can be used.

4) IP address information to support TCP LRP-DT instances. Either, both, or neither of the addresses can be supplied. If either or both, a TCP port number is supplied.

   i) An IPv4 address for this system.

   ii) An IPv6 address for this system.

   iii) A TCP port number to be used for passive TCP OPENs (6.9.2.3).

NOTE 2—If no IP address is given, TCP is not enabled. If neither ECP nor TCP are enabled, the Local Target Port request is a delete operation.

NOTE 3—See B.6 for a discussion of the choice between ECP and TCP.

c) LRP-DS Hello parameters to be used by the Portal state machines (8.2.4, 8.2.5) created by this request:

   1) Hello Time (9.4.2.5).

   2) Information for the Application Information TLV (9.4.2.10).

   3) A value for the cplCompleteListTimerReset variable (8.4.6.3).

10.2.2.2 Outputs

a) Status:

   1) **badInput:** Invalid inputs, e.g., the specified port does not exist.

   2) **success:** Portal creation (or destruction) is in progress. A Portal status indication (10.2.6) will be generated to indicate the results of the Local Target Port request.

10.2.3 Neighbor Target Port request

Neighbor Target Port request is used to create or delete neighbor target ports, and thus create or destroy LRP-DT instance and Portals for the issuing application instance. A Neighbor Target Port request creates or deletes an entry in the imPplNeighborList (item g in 7.2.2.1) of a particular entry in the imTargetPortList. See 7.2.3.2 for a full description of the actions taken when a Neighbor Target Port request is issued.

A system shall not accept a Neighbor Target Port request that specifies a Chassis ID and Port ID match that is in another Neighbor Target Port request for the same entry in the imPplNeighborList.

10.2.3.1 Inputs

a) Request identification parameters:

   1) A reference to an entry in the imTargetPortList (7.2.2.1), corresponding to a previously issued Local Target Port request (10.2.2).

b) Neighbor LLDP parameters:

NOTE 1—If the IEEE 802.1AB Link Layer Discovery Protocol (LLDP) is used, these are the parameters received from a particular LLDP neighbor by the Native or Controlled system on the selected target port:

   1) The Chassis ID of the neighbor system's target port (8.5.2 of IEEE Std 802.1AB-2016) or null.

   2) The Port ID of the neighbor system's target port (8.5.3 of IEEE Std 802.1AB-2016) or null.

   3) ECP address information for the neighbor system to support ECP LRP-DT instances. Either, all, or none of the following:

      i) Either an ECP MAC address for the neighbor system, or null, indication that the LRP-DT ECP mechanism (6.9.1) cannot be used.

      ii) A Boolean value to be placed in the imPplExploreXmit flag (item h:4 in 7.2.2.1).

iii) A Boolean value to be placed in the imPplExploreRecv flag (item h:5 in 7.2.2.1).

4) IP address information for the neighbor system to support TCP LRP-DT instances. Either, both, or neither of the addresses can be supplied. If either or both, a TCP port number is supplied

i) An IPv4 address for this system.

ii) An IPv6 address for this system.

iii) A TCP Port number for the LRP Proxy instance for this system.

NOTE 2—If no IP address is given, TCP is not enabled. If neither ECP nor TCP address information is supplied, the Neighbor Target Port request is a delete operation.

10.2.3.2 Outputs

a) Status:

1) **badInput:** Invalid inputs, e.g., the specified imTargetPortList entry does not exist. See also 7.2.2.1 for a list of conditions that would lead to an invalid imTargetPortList, and thus return the badInput code.

2) **success:** Portal creation (or destruction) is in progress. A Portal status indication (10.2.6) will be generated to indicate the results of the Local Target Port request.

10.2.4 Associate Portal request

The Associate Portal request is issued by the application instance in response to a First Hello indication (10.2.5). The Associate Portal request triggers a call to pamAssociatePortal (8.2.3.1).

10.2.4.1 Inputs

a) PortalId

b) Boolean indicating whether the Portal association is allowed (TRUE) or not (FALSE).

10.2.4.2 Outputs

a) Status:

1) **success:** Operation successful. Portal will continue the Hello LRPDU handshake, which can result in a Portal Association.

2) **badInputs:** Invalid inputs, Link-local Registration Protocolsunknown PortalId, or Associate Portal request out of sequence.

10.2.5 First Hello indication

The First Hello indication is generated when the first Hello LRPDU is received from the neighbor Portal. The application instance can check the information supplied in the First Hello indication and respond with a Associate Portal request (10.2.4) to either create or destroy the Portal.

10.2.5.1 Outputs

a) PortalId. This PortalId can be used in subsequent Associate Portal requests (10.2.4) and Portal status indications (10.2.6) to refer to this same Portal.

b) The contents of the received Hello LRPDU.

10.2.6 Portal status indication

10.2.6.1 Outputs

a) PortalId
b) Association status:
    1) **connected:** Portal association is complete.
    2) **disconnected:** During or after Portal association, either a Hello LRPDU was received with a Hello status indicating that it is reinitializing the connection, or no Hello LRPDU has been received since the time in the last-received Hello Time (9.4.2.5) expired.
c) Neighbor Registrar Database Overflow: A Boolean, equal to the value of pamNeighborOverflow (8.2.2.11).
d) Neighbor Registrar Database Correct: A Boolean, equal to the value of pamNeighborAcknowledged (8.2.2.12).

10.3 Portal interface

10.3.1 Applicant database primitives

10.3.1.1 Write record request

A Write record request can create a new record, or alter or delete an existing record. If no record with the specified record number (item b in 10.3.1.1.1) exists and the size of the data (item c in 10.3.1.1.1) is non-zero, a new record is created. If the record exists, its contents are altered, and if the data size is 0, the record is deleted.

NOTE—There is no provision for what happens if the applicant database, ostensibly maintained by LRP-DS, overflows its allocable memory. The API is not visible outside the box, and it is not clear that passing that error status to the registrar is of any use.

10.3.1.1.1 Inputs

a) PortalId
b) Record number (item b.1 in 6.8)
c) Data

10.3.1.1.2 Outputs

a) Status:
    1) **success:** Local applicant database changed, state machines will try to propagate the change.
    2) **badInputs:** Invalid inputs, e.g., unknown PortalId.

10.3.2 Registrar database primitives

10.3.2.1 Delete record request

This request calls regDeleteRecord (8.4.4.3) to delete a record from the registrar database.

NOTE—The purpose of this primitive is to allow the application instance above the registrar database to take appropriate action when a Portal status indication reports that the Portal association has been interrupted. This could be used, for example, to delete the entire registrar database in anticipation of a possible re-association. The consequences of using the Delete record request during the normal operation of LRP are not examined in this standard.

10.3.2.1.1 Inputs

a) PortalId.
b) Record number to be deleted.

10.3.2.2 Database overflow request

This request writes the pamLocalOverflow (8.2.2.10) Boolean for the Partial list state machine (8.4.5) indicated by the PortalId (item a in 10.3.2.2.1). This triggers the Send Hello state machines to transmit Hello LRPDUs with the database overflow bit set in the Error status field (9.4.2.3) of transmitted Hello LRPDUs.

10.3.2.2.1 Inputs

a) PortalId.
b) Boolean indicating whether the application instance's registrar database is (TRUE) or is not (FALSE) in an overflow condition.

10.3.2.2.2 Outputs

a) Status
   1) **success:** Status changed locally, state machines will try to propagate the change.
   2) **badInputs:** Invalid inputs, e.g., unknown PortalId.

10.3.2.3 Record written indication

This indication is produced when the Partial list state machine (8.4.5) receives a Record LRPDU (9.4.3) that does not match its cached record header information; that is, when a record is created, changed, or deleted. If the data in the Record written indication (item c in 10.3.2.3.1) has zero length, then the record has been deleted.

10.3.2.3.1 Outputs

a) PortalId
b) Record number (item b.1 in 6.8)
c) Data

11. Managed objects

11.1 Introduction

Managed objects for monitoring LRP-DS Portals (Clause 8) are specified in 11.4, and for monitoring LRP-DT instances (7.3) in 11.5.

The managed objects in 11.4 and 11.5 can only monitor Portals and LRP-DT instances; no managed objects are provided that explicitly create or manage them. Instead, primitives are provided in Clause 10 for an application instance to manage the Portals and LRP-DT instances. This standard assumes that the specification of a particular LRP application would supply managed objects for creating and managing Portals and/or their LRP-DT instances, if such objects are necessary.

11.2 Managed objects UML

A UML model for the managed objects for the LRP is given in Figure 2-1, using the language of OMG UML Version 2.5.

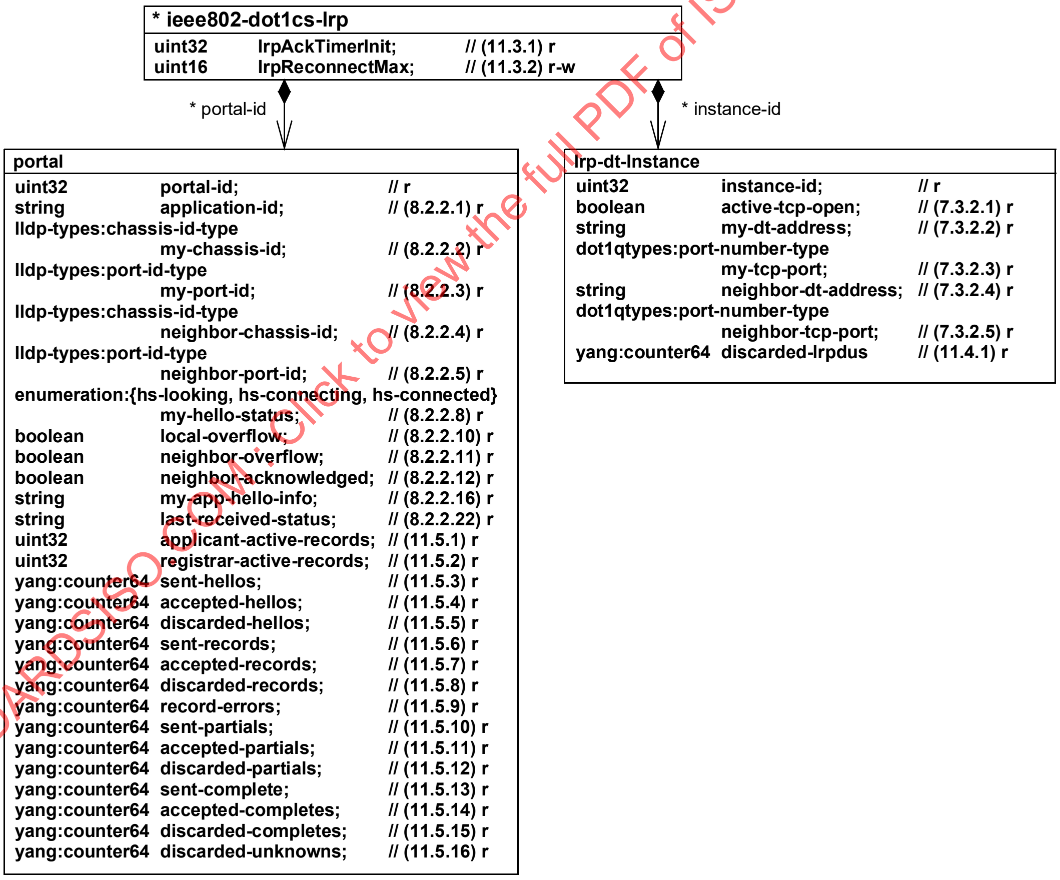

* ieee802-dot1cs-lrp
uint32 lrpAckTimerInit; // (11.3.1) r
uint16 lrpReconnectMax; // (11.3.2) r-w
* portal-id
* instance-id
portal
uint32 portal-id; // r
string application-id; // (8.2.2.1) r
lldp-types:chassis-id-type my-chassis-id; // (8.2.2.2) r
lldp-types:port-id-type my-port-id; // (8.2.2.3) r
lldp-types:chassis-id-type neighbor-chassis-id; // (8.2.2.4) r
lldp-types:port-id-type neighbor-port-id; // (8.2.2.5) r
enumeration:{hs-looking, hs-connecting, hs-connected} my-hello-status; // (8.2.2.8) r
boolean local-overflow; // (8.2.2.10) r
boolean neighbor-overflow; // (8.2.2.11) r
boolean neighbor-acknowledged; // (8.2.2.12) r
string my-app-hello-info; // (8.2.2.16) r
string last-received-status; // (8.2.2.22) r
uint32 applicant-active-records; // (11.5.1) r
uint32 registrar-active-records; // (11.5.2) r
yang:counter64 sent-hellos; // (11.5.3) r
yang:counter64 accepted-hellos; // (11.5.4) r
yang:counter64 discarded-hellos; // (11.5.5) r
yang:counter64 sent-records; // (11.5.6) r
yang:counter64 accepted-records; // (11.5.7) r
yang:counter64 discarded-records; // (11.5.8) r
yang:counter64 record-errors; // (11.5.9) r
yang:counter64 sent-partials; // (11.5.10) r
yang:counter64 accepted-partials; // (11.5.11) r
yang:counter64 discarded-partials; // (11.5.12) r
yang:counter64 sent-complete; // (11.5.13) r
yang:counter64 accepted-completes; // (11.5.14) r
yang:counter64 discarded-completes; // (11.5.15) r
yang:counter64 discarded-unknowns; // (11.5.16) r
lrp-dt-Instance
uint32 instance-id; // r
boolean active-tcp-open; // (7.3.2.1) r
string my-dt-address; // (7.3.2.2) r
dot1qtypes:port-number-type my-tcp-port; // (7.3.2.3) r
string neighbor-dt-address; // (7.3.2.4) r
dot1qtypes:port-number-type neighbor-tcp-port; // (7.3.2.5) r
yang:counter64 discarded-lrpdus // (11.4.1) r

Figure 11-1—UML model for LRP managed objects

The Per-LRP-DT instance managed objects (11.4) are shown as the lrp-dt-Instance table. The table of Per-Portal managed objects (11.5) is shown in Figure 2-1 as the portal table.

The method used to index entries in the tables is expanded from the simple "reference" terminology (1.5) used for the state machine descriptions.

11.3 System global managed objects

11.3.1 lrpAckTimerInit

A read-only integer specifying the number of milliseconds for ackTimerInit (D.2.12.6 of IEEE Std 802.1Q-2018).

11.3.2 lrpReconnectMax

An integer number of seconds that is the maximum value for instReconnectReset (7.3.2.7).

NOTE—If a network controller that is a Proxy for number of relay systems is down when the network is turned on, then the Native end systems' active TCP connections will fail. Gradually, their attempts to reconnect will slow down. When the network controller comes up, those slow retries will take some time to complete. Such scenarios need to be accounted for when setting a value for lrpReconnectMax.

11.4 Per-LRP-DT instance managed objects

There exists one set of managed objects for each LRP-DT instance. These objects are created and/or destroyed as described in 7.2 and 8.2.1.

All of the Per-LRP-DT instance managed objects are read-only. There are two kinds of Per-LRP-DT instance managed objects:

a) Objects that correspond directly to a state machine variables in 7.3.2; and
b) Counter objects that count events on the LRP-DT instance.

The Per-LRP-DT instance managed objects are listed in Table 11-1.

Table 11-1—Per-LRP-DT instance managed objects

Managed object	Reference	MIB object (13.5.2)
instActiveTcp	7.3.2.1	lrpDtInstActiveTcp
instMyAddress	7.3.2.2	lrpDtInstMyAddress
instMyPortNumber	7.3.2.3	lrpDtInstMyTcpPort
instNeighborAddress	7.3.2.4	lrpDtInstNeighborAddress
instNeighborPortNumber	7.3.2.5	lrpDtInstNeighborTcpPort
instDiscardedLRPDUs	11.4.1	lrpDtInstDiscardedLrpdus

11.4.1 instDiscardedLRPDUs

A counter indicating the number of LRPDUs discarded by the LRP-DT instance that cannot be assigned to a Portal for processing. See 7.3.3.1.

11.5 Per-Portal managed objects

There exists one set of managed objects for each Portal that can potentially be created in a system. These objects are created and/or destroyed as described in 8.2.1.

All of the Per-Portal managed objects are read-only.

All of the Per-Portal managed objects are read-only. There are three kinds of Per-Portal managed objects, as follows:

a) Objects that correspond directly to a state machine variables in Clause 8.
b) Portal status objects that are not in Clause 8.
c) Counter objects that count events on the Portal.

The Per-Portal managed objects comprising each set are listed in Table 11-2.

Table 11-2—Per-Portal managed objects

Managed object	Reference	MIB object (13.5.2)
pamAppId	8.2.2.1	lrpPortalAppId
pamMyChassisId	8.2.2.2	lrpPortalMyChassisIdType, lrpPortalMyChassisId
pamMyPortId	8.2.2.3	lrpPortalMyPortIdType, lrpPortalMyPortId
pamNeighborChassisId	8.2.2.4	lrpPortalNbrChassisIdType, lrpPortalNbrChassisId
pamNeighborPortId	8.2.2.5	lrpPortalNbrPortIdType, lrpPortalNbrPortId
pamLocalOverflow	8.2.2.10	lrpPortalLocalOverflow
ApplicantActiveRecords	11.5.1	lrpPortalApplicantActiveRecords
RegistrarActiveRecords	11.5.2	lrpPptCtRegistrarActiveRecords
SentHellos	11.5.3	lrpPptCtSentHellos
AcceptedHellos	11.5.4	lrpPptCtAcceptedHellos
DiscardedHellos	11.5.5	lrpPptCtDiscardedHellos
SentRecords	11.5.6	lrpPptCtSentRecords
AcceptedRecords	11.5.7	lrpPptCtAcceptedRecords
DiscardedRecords	11.5.8	lrpPptCtDiscardedRecords
RecordErrors	11.5.9	lrpPptCtRecordErrors
SentPartials	11.5.10	lrpPptCtSentPartials
AcceptedPartials	11.5.11	lrpPptCtAcceptedPartials
DiscardedPartials	11.5.12	lrpPptCtDiscardedPartials
SentCompletes	11.5.13	lrpPptCtSentCompletes

Table 11-2—Per-Portal managed objects *(continued)*

Managed object	Reference	MIB object (13.5.2)
AcceptedCompletes	11.5.14	lrpPptCtAcceptedCompletes
DiscardedCompletes	11.5.15	lrpPptCtDiscardedCompletes
DiscardedUnknowns	11.5.16	lrpPptCtDiscardedUnknowns

11.5.1 ApplicantActiveRecords

An integer reporting the number of records in the applicant database 8.3.2.

11.5.2 RegistrarActiveRecords

An integer reporting the number of records in the registrar database 8.4.2.

11.5.3 SentHellos

The number of Hello LRPDUs (9.4.2) transmitted by the Send Hello state machines (8.2.4).

11.5.4 AcceptedHellos

The number of valid Hello LRPDUs (9.4.2) received by the Receive Hello state machine (8.3.4).

11.5.5 DiscardedHellos

The number of invalid Hello LRPDUs (9.4.2) discarded by the Receive Hello state machine (8.3.4).

11.5.6 SentRecords

The number of Record LRPDUs (9.4.3) transmitted by the Applicant state machine (8.3.4).

11.5.7 AcceptedRecords

The number of valid Record LRPDUs (9.4.3) received by the Partial list state machine (8.4.5).

11.5.8 DiscardedRecords

The number of invalid Record LRPDUs (9.4.3) discarded by the Partial list state machine (8.4.5).

11.5.9 RecordErrors

The number of records discarded from otherwise-valid Record LRPDUs (9.4.3) by regReceiveWriteRecord (8.4.4.1) due to inconsistencies between the Checksum, Application data, and Data length fields.

11.5.10 SentPartials

The number of Partial List LRPDUs (9.4.4) transmitted by the Applicant state machine (8.3.4).

11.5.11 AcceptedPartials

The number of valid Partial List LRPDUs (9.4.4) received by the Applicant state machine (8.3.4).

11.5.12 DiscardedPartials

The number of invalid Partial List LRPDUs (9.4.4) discarded by the Applicant state machine (8.3.4).

11.5.13 SentCompletes

The number of Complete List LRPDUs (9.4.5) transmitted by the Applicant state machine (8.3.4).

11.5.14 AcceptedCompletes

The number of valid Complete List LRPDUs (9.4.5) received by the Applicant state machine (8.3.4).

11.5.15 DiscardedCompletes

The number of invalid Complete List LRPDUs (9.4.5) discarded by the Applicant state machine (8.3.4).

11.5.16 DiscardedUnknowns

The number of LRPDUs of unknown type discarded by the Applicant state machine (8.3.4) or Partial list state machine (8.4.5).

11.6 LRP LLDP TLV managed objects

There are managed objects for the IEEE 802.1AB LLDP organization-specific TLVs transmitted and received by a system, the LRP ECP Discovery TLV (C.2.1) and the LRP TCP Discovery TLV (C.2.2). These managed objects fall into two classes:

a) Information about a Proxy system's IP addresses has to be known to a Controlled system for that Controlled system to be able to transmit LRP TCP Discovery TLVs that advertise the Proxy system's IP addresses. This constitutes the only functionality that must be added to a Controlled system in order for that system to have LRP run on its behalf by a Proxy system (11.6.1).

b) Any Native system or Controlled system can make use of the set of managed objects that enable a management system or Proxy system to observe the contents of the LRP ECP Discovery TLVs and LRP TCP Discovery TLVs being transmitted and received by the system (11.6.2).

11.6.1 Controlled system IP address information

11.6.1.1 lrpLldpTcpIpAddressses

There is zero or one instance of the lrpLldpTcpIpAddressses object for each LRP application running in a system. This object contains a TCP Port number, zero or one IPv4 address, zero or one IPv6 address, and a Boolean for each IP address, indicating whether or not this information is to be advertised by the system in LRP TCP Discovery TLVs (C.2.2).

11.6.2 LRP LLDP TLV information

11.6.2.1 lrpLldpEcpXmitEnable

There is zero or one instance of this Boolean value for each LRP application, for each port, for each LLDP instance. If TRUE, the LRP ECP Discovery TLV (C.2.1) is to be transmitted on that port and LLDP instance with that LRP application's AppId. If not present or FALSE, the LRP ECP Discovery TLV it not transmitted for that port and LLDP instance.

11.6.2.2 lrpLldpEcpTlvXmitInfo

There is zero or one entry in this table for each LRP application, for each port, for each LLDP instance. Each entry contains the AppId and MAC address to advertise in an LRP ECP Discovery TLV (C.2.1), if enabled by a corresponding lrpLldpEcpXmitEnable (11.6.2.1).

11.6.2.3 lrpLldpEcpTlvRecvInfo

There is zero or one entry in this table for each LRP application, for each port, for each LLDP instance, for each LLDP neighbor. Each entry contains the AppId and MAC address received from an LRP ECP Discovery TLV (C.2.1).

11.6.2.4 lrpLldpTcpXmitEnable

There is zero or one instance of this Boolean value for each LRP application, for each port, for each LLDP instance. If TRUE, the LRP TCP Discovery TLV (C.2.2) is to be transmitted on that port and LLDP instance with that LRP application's AppId. If not present or FALSE, the LRP TCP Discovery TLV it not transmitted for that port and LLDP instance.

11.6.2.5 lrpLldpTcpTlvXmitInfo

There is zero or one entry in this table for each LRP application, for each port, for each LLDP instance. Each entry contains the information to advertise in an LRP TCP Discovery TLV (C.2.2), if enabled by a corresponding lrpLldpTcpXmitEnable (11.6.2.4).

11.6.2.6 lrpLldpTcpTlvRecvInfo

There is zero or one entry in this table for each LRP application, for each port, for each LLDP instance, for each LLDP neighbor. Each entry contains the information received from an LRP TCP Discovery TLV (C.2.2).

12. YANG models for LRP

12.1 Introduction

This clause specifies YANG data modules that provide status monitoring of the LRP. No configuration capabilities are present in these modules; configuration is expected to be performed at the LRP application layer.

The YANG data modules are based on the UML model for LRP managed objects in 11.2. The YANG data model has not been derived from the MIB, and there has been no attempt to include data or modeling constructs that might appear in the MIB but not in the information model.

12.2 The YANG framework

This clause has been developed according to the YANG guidelines published in IETF RFC 6087 [B13] as applicable to IEEE standards. The YANG framework applies hierarchy in the following areas:

1) The uniform resource name (URN), as specified in IEEE Std 802d™ [B2]. The structure of the URN is such that “ieee” is the root (i.e., name-space identifier), followed by the standard, then the working group developing the standard.
2) The YANG objects form a hierarchy of configuration and operational data structures that define the YANG model. These hierarchical relationships are described in 11.2 and 12.5.

12.3 Security considerations

The YANG modules defined in this clause are designed to be accessed via a network configuration protocol (e.g., NETCONF protocol, IETF RFC 6241 [B14]). In the case of NETCONF, the lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH (IETF RFC 4251 [B10]). The NETCONF access control model (IETF RFC 8341 [B17]) provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

It is the responsibility of a system’s implementor and administrator to ensure that the protocol entities in the system that support NETCONF, and any other remote configuration protocols that make use of these YANG modules, are properly configured to allow access only to those principals (users) that have legitimate rights to read or write data nodes. This standard does not specify how the credentials of those users are to be stored or validated.

The YANG module in 12.5 and 12.6 does not contain any configurable items. The security threats are therefore limited to observation of the operation of LRP. Such read-only objects may be considered sensitive or vulnerable in some network environments.

12.4 Relationship to other YANG modules

The LRP YANG module augments the /sys:system module, so is applicable to most systems. No YANG module corresponding to the LLDP extensions specified in 13.5.3 is provided.

12.5 YANG data scheme definition

```
module: ieee802-dot1cs-lrp
  augment /sys:system:
   +--rw lrp {lrp}?
      +--ro lrp-ack-timer-init?   uint32
      +--rw lrp-reconnect-max?    uint16
      +--ro portal* [portal-id]
      |  +--ro portal-id                   uint32
      |  +--ro application-id?             string
      |  +--ro my-chassis-id?              lldp-types:chassis-id-type
      |  +--ro my-port-id?                 lldp-types:port-id-type
      |  +--ro neighbor-chassis-id?        lldp-types:chassis-id-type
      |  +--ro neighbor-port-id?           lldp-types:port-id-type
      |  +--ro my-hello-status?            enumeration
      |  +--ro local-overflow?             boolean
      |  +--ro neighbor-overflow?          boolean
      |  +--ro neighbor-acknowledged?      boolean
      |  +--ro my-app-hello-info?          string
      |  +--ro last-received-status?       string
      |  +--ro applicant-active-records?   uint32
      |  +--ro registrar-active-records?   uint32
      |  +--ro sent-hellos?                yang:counter64
      |  +--ro accepted-hellos?            yang:counter64
      |  +--ro discarded-hellos?           yang:counter64
      |  +--ro sent-records?               yang:counter64
      |  +--ro accepted-records?           yang:counter64
      |  +--ro discarded-records?          yang:counter64
      |  +--ro record-errors?              yang:counter64
      |  +--ro sent-partials?              yang:counter64
      |  +--ro accepted-partials?          yang:counter64
      |  +--ro discarded-partials?         yang:counter64
      |  +--ro sent-complete?              yang:counter64
      |  +--ro accepted-completes?         yang:counter64
      |  +--ro discarded-completes?        yang:counter64
      |  +--ro discarded-unknowns?         yang:counter64
      +--ro lrp-dt-instance* [instance-id]
         +--ro instance-id              uint32
         +--ro active-tcp-open?         boolean
         +--ro my-dt-address?           string
         +--ro my-tcp-port?             dot1qtypes:port-number-type
         +--ro neighbor-dt-address?     string
         +--ro neighbor-tcp-port?       dot1qtypes:port-number-type
         +--ro discarded-lrpdus?        yang:counter64
```

12.6 Definition of LRP YANG module[14], [15]

```
module ieee802-dot1cs-lrp {
  yang-version 1.1;

  namespace "urn:ieee:std:802.1CS:yang:ieee802-dot1cs-lrp";
  prefix "dot1cs";

  import ietf-system { prefix "sys"; }
  import ieee802-dot1ab-types { prefix "lldp-types"; }
  import ietf-yang-types { prefix "yang"; }
```

[14]Copyright release for YANG module: Users of this standard may freely reproduce the YANG module contained in this subclause so that they can be used for their intended purpose.

[15]An ASCII version of each YANG module is attached to the PDF of this amendment and can also be obtained from the IEEE 802.1 Website at https://1.ieee802.org/yang-modules/.

```
import ieee802-dot1q-types { prefix dot1qtypes; }
import ietf-interfaces { prefix if; }
import ieee802-types { prefix ieee; }
import ietf-inet-types { prefix inet; }

organization
  "Institute of Electrical and Electronics Engineers";
contact
  "WG-URL: http://ieee802.org/1/
   WG-EMail: stds-802-1-l@ieee.org
     Contact: IEEE 802.1 Working Group Chair
     Postal: C/O IEEE 802.1 Working Group
     IEEE Standards Association
          445 Hoes Lane
          Piscataway
          NJ 08854
          USA

  E-mail: stds-802-1-chairs@ieee.org";

description
  "Managed objects to control the Link-local Registration Protocol,
   IEEE Std 802.1CS";

   Copyright (C) IEEE (2021). This version of this YANG module
   is included in Clause 12 of IEEE Std 802.1CS-2020;
   see the standard itself for full legal notices. ";

 revision 2020-12-03 {
  description
    "First defined in IEEE P802.1CS-2020";
  reference
    "IEEE Std 802.1CS-2020.";
}
/*---------------------*/
/* Feature             */
/*---------------------*/

feature lrp {
  description
    "Feature Link-local Registration Protocol";
}

/*---------------------*/
/* Type Definitions    */
/*---------------------*/
typedef lrp-dt-address-union {
  type union {
    type ieee:mac-address;
    type inet:ipv4-address;
    type inet:ipv6-address;
  }
}

/*---------------------*/
/* Configuration Data */
/*---------------------*/
/*
```

```
    Link-local Registration Protocol
  */
  augment "/sys:system" {
    description "Link-local Registration Protocol";
    container lrp {
      if-feature lrp;
      description
        "Configure the Link-local Registration Protocol";
        leaf lrp-ack-timer-init {
          type uint32;
          units "milliseconds";
          config false;
          description
            "A read-only integer n specifying the number of milliseconds for
            ackTimerInit (D.2.12.6 of IEEE Std 802.1Q-2018)";
          reference
            "IEEE 802.1CS Clause 11.3.1";
        }
        leaf lrp-reconnect-max {
          type uint16;
          units "seconds";
          description
            "An integer number of seconds which is the maximum value for
            instReconnectReset.";
          reference
            "IEEE 802.1CS Clause 11.3.2";
        }
        list portal {
        key "portal-id";
        config false;
        leaf portal-id {
          type uint32;
          config false;
          description
            "Local Identifier of portal";
          reference
            "IEEE 802.1CS Clause 10";
        }
        leaf target-port-interface-ref {
          type if:interface-ref;
          config false;
          description
            "The interface reference identifying the target
            port to which this portal is attached to.";
          reference
            "IEEE 802.1CS Clause (8.2.2.1)";
        }
        leaf lrp-dt-instance-id {
          type leafref {
            path "/sys:system/dot1cs:lrp/dot1cs:lrp-dt-instance/dot1cs:instance-
id";
          }
          config false;
          description
            "The LRP-DT instance id that this portal is
            attached to.";
          reference
            "IEEE 802.1CS Clause (8.2.2.1)";
        }
```

```
leaf application-id {
  type string;
  config false;
  description
    "The value transmitted in, and expected to be received in, the
    appId field of Hello LRPDUs.";
  reference
    "IEEE 802.1CS Clause (8.2.2.1)";
}
leaf my-chassis-id {
  type lldp-types:chassis-id-type ;
  config false;
  description
    "The value transmitted in the My Chassis ID TLV, and expected to be
    received in the Neighbor Chassis ID TLV, of Hello LRPDUs";
  reference
    "IEEE 802.1CS Clause (8.2.2.2)";
}
leaf my-port-id {
  type lldp-types:port-id-type ;
  config false;
  description
    "The value transmitted in the My Port ID TLV, and expected to be
    received in the Neighbor Port ID TLV, of Hello LRPDUs.";
  reference
    "IEEE 802.1CS Clause (8.2.2.3)";
}
leaf neighbor-chassis-id {
  type lldp-types:chassis-id-type ;
  config false;
  description
    "The value transmitted in the Neighbor Chassis ID TLV, and expected
    to be received in the My Chassis ID TLV, of Hello LRPDUs.";
  reference
    "IEEE 802.1CS Clause (8.2.2.4)";
}
leaf neighbor-port-id {
  type lldp-types:port-id-type ;
  config false;
  description
    "The value transmitted in the Neighbor Port ID TLV, and expected to
    be received in the My Port ID TLV, of Hello LRPDUs.";
  reference
    "IEEE 802.1CS Clause (8.2.2.5)";
}
leaf my-hello-status {
  type enumeration {
    enum hs-looking {
      value 1;
      description
        "This Portal has not yet received a successful Associate Portal
        request.";
    }
    enum hs-connecting {
      value 2;
      description
        "This Portal has received a successful Associate Portal
        request, and a Hello LRPDU with the hsLooking status. The
        Portal is ready to receive all LRPDUs.";
      }
      enum hs-connected {
        value 3;
        description
          "This Portal is up and ready to transfer LRP application data.
          The Portal is allowed to transmit all LRPDUs.";
      }
    }
    config false;
    description
      "An enumerated value to be transmitted in the Hello status field
      of any Hello LRPDU.";
    reference
      "IEEE 802.1CS Clause (8.2.2.8)";
  }
  leaf local-overflow {
    type boolean;
    config false;
    description
      "Contains the last Boolean input from the Database overflow
      request. A value of TRUE indicates that the partner applicant
      database has exceeded the capacity of the local registrar LRP
      application.";
    reference
      "IEEE 802.1CS Clause (8.2.2.10)";
  }
  leaf neighbor-overflow {
    type boolean;
    config false;
    description
      "A Boolean copied from the last-received database overflow bit [bit
      8] in the Error status field of the last-received Hello LRPDU.";
    reference
      "IEEE 802.1CS Clause (8.2.2.11)";
  }
  leaf neighbor-acknowledged {
    type boolean;
    config false;
    description
      "A Boolean, equal to the AND of all of the actAcknowledged
      variables for all of the Applicant state machines (records) on this
      Portal.";
    reference
      "IEEE 802.1CS Clause (8.2.2.12)";
  }
  leaf my-app-hello-info {
    type string;
    config false;
    description
      "Value to put in the Application Information TLV of a
      transmitted Hello LRPDU.";
    reference
      "IEEE 802.1CS Clause (8.2.2.16)";
  }
  leaf last-received-status {
    type string;
    config false;
    description
      "Used by the Receive Hello state machine to record the
```

```
              Hello status field of a Hello LRPDU received from the
              neighbor Portal.";
            reference
              "IEEE 802.1CS Clause (8.2.2.22)";
          }
          leaf applicant-active-records {
            type uint32;
            config false;
            description
              "An integer reporting the number of records in the applicant
              database.";
            reference
              "IEEE 802.1CS Clause (11.5.1)";
          }
          leaf registrar-active-records {
            type uint32;
            config false;
            description
              "An integer reporting the number of records in the registrar
              database.";
            reference
              "IEEE 802.1CS Clause (11.5.2)";
          }
          leaf sent-hellos {
            type yang:counter64;
            config false;
            description
                 "The number of Hello LRPDUs transmitted by the Send Hello state
machines.";
            reference
              "IEEE 802.1CS Clause (11.5.3)";
          }
          leaf accepted-hellos {
            type yang:counter64;
            config false;
            description
              "The number of valid Hello LRPDUsreceived by the Receive Hello
              state machine.";
            reference
              "IEEE 802.1CS Clause (11.5.4)";
          }
          leaf discarded-hellos {
            type yang:counter64;
            config false;
            description
              "The number of invalid Hello LRPDUs discarded by the Receive Hello
              state machine";
            reference
              "IEEE 802.1CS Clause (11.5.5)";
          }
          leaf sent-records  {
            type yang:counter64;
            config false;
            description
              "The number of Record LRPDUs transmitted by the Applicant state
              machine.";
            reference
              "IEEE 802.1CS Clause (11.5.6)";
          }
```

```
leaf accepted-records {
  type yang:counter64;
  config false;
  description
    "The number of valid Record LRPDUs received by the Partial list
    state machine.";
  reference
    "IEEE 802.1CS Clause (11.5.7)";
}
leaf discarded-records {
  type yang:counter64;
  config false;
  description
    "The number of invalid Record LRPDUs discarded by the Partial list
    state machine.";
  reference
    "IEEE 802.1CS Clause (11.5.8)";
}
leaf record-errors  {
  type yang:counter64;
  config false;
  description
    "The number of records discarded from otherwise-valid Record LRPDUs
    by regReceiveWriteRecord due to inconsistencies between the
    Checksum, Application data, and Data length fields.";
  reference
    "IEEE 802.1CS Clause (11.5.9)";
}
leaf sent-partials  {
  type yang:counter64;
  config false;
  description
    "The number of Partial List LRPDUs transmitted by the Applicant
    state machine.";
  reference
    "IEEE 802.1CS Clause (11.5.10)";
}
leaf accepted-partials {
  type yang:counter64;
  config false;
  description
    "The number of valid Partial List LRPDUs received by the Applicant
    state machine.";
  reference
    "IEEE 802.1CS Clause (11.5.11)";
}
leaf discarded-partials {
  type yang:counter64;
  config false;
  description
    "The number of invalid Partial List LRPDUs discarded by the
    Applicant state machine.";
  reference
    "IEEE 802.1CS Clause (11.5.12)";
}
leaf sent-complete  {
  type yang:counter64;
  config false;
  description
```

```
          "The number of Complete List LRPDUs transmitted by the Applicant
          state machine.";
        reference
          "IEEE 802.1CS Clause (11.5.13)";
      }
      leaf accepted-completes {
        type yang:counter64;
        config false;
        description
          "The number of valid Complete List LRPDUs received by the Applicant
          state machine.";
        reference
          "IEEE 802.1CS Clause (11.5.14)";
      }
      leaf discarded-completes{
        type yang:counter64;
        config false;
        description
          "The number of invalid Complete List LRPDUs discarded by the
          Applicant state machine.";
        reference
          "IEEE 802.1CS Clause (11.5.15)";
      }
      leaf discarded-unknowns {
        type yang:counter64;
        config false;
        description
          "The number of LRPDUs of unknown type discarded by the Applicant
          state machine or Partial list state machine.";
        reference
          "IEEE 802.1CS Clause (11.5.16)";
      }
    } // end portal
    list lrp-dt-instance {
    key "instance-id";
    config false;
    leaf instance-id {
      type uint32;
      config false;
      description
        "Local data transport instance";
      reference
        "IEEE 802.1CS Clause 7";
      }
      leaf active-tcp-open {
        type boolean;
        config false;
        description
          "A Boolean value that is TRUE if and only if instMyAddress and
          instNeighborAddress are TCP addresses and this LRP-DT instance is
          using the active , not the passive , form of TCP OPEN";
        reference
          "IEEE 802.1CS Clause (7.3.2.1)";
      }
      leaf my-dt-address {
        type lrp-dt-address-union;
        config false;
        description
          "The address of the local system for this LRP-DT instance; the
```

```
            address used as a destination address by the neighbor LRP-DT
            instance. The address includes a type (MAC, IPv4, or IPv6) and an
            address of that type.";
          reference
            "IEEE 802.1CS Clause (7.3.2.2)";
        }
        leaf my-tcp-port{
          type inet:port-number;
          config false;
          description
            "The local port number for this TCP connection, or 0, if this
            connection uses ECP, instead of TCP.";
          reference
            "IEEE 802.1CS Clause (7.3.2.3)";
        }
        leaf neighbor-dt-address {
          type lrp-dt-address-union;
          config false;
          description
            "The address of the neighbor LRP-DT instance; the address used as a
            destination address by this LRP-DT instance. The address includes a
            type (MAC, IPv4, or IPv6) and an address of that type.";
          reference
            "IEEE 802.1CS Clause (7.3.2.4)";
        }
        leaf neighbor-tcp-port {
          type inet:port-number;
          config false;
          description
            "The remote port number for this TCP connection, or 0, if this
            connection uses ECP, instead of TCP.";
          reference
            "IEEE 802.1CS Clause (7.3.2.5)";
        }
        leaf discarded-lrpdus {
          type yang:counter64;
          config false;
          description
            "A counter indicating the number of LRPDUs discarded by the LRP-DT
            instance that cannot be assigned to a Portal for processing.";
          reference
            "IEEE 802.1CS Clause (11.4.1)";
        }
      } // end lrp-dt-instance
    } // end lrp
  } // end augment system
} // end ieee802-dot1cs-lrp
```

13. MIB modules for LRP[16]

13.1 Internet standard management framework

The clause contains a complete SMIv2 MIB set for all features of this standard.

For a detailed overview of the documents that describe the current Internet Standard Management Framework, refer to section 7 of IETF RFC 3410 (2002). Managed objects are accessed via a virtual information store, termed the *Management Information Base* or *MIB*. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This clause specifies a MIB module that is compliant to the SMIv2, which is described in IETF STD 58, IETF RFC 2578, IETF RFC 2579, and IETF RFC 2580.

13.2 Structure of the LRP MIB

The IEEE 802.1CS MIB is divided into three modules, as shown in Table 13-1.

Table 13-1—Structure of the MIB modules

Module	Subclause	Reference	Notes
LRP-TC-MIB	13.5.1	Clause 11	Textual conventions used by the LRP-MIB
LRP-MIB	13.5.2	Clause 11	The LRP MIB
LLDP-V2-LRP-EXT-MIB	13.5.3	Annex C	LRP extensions to the LLDP MIBs

13.2.1 Structure of the LRP-TC-MIB

The purpose of the LRP-TC-MIB is to define textual conventions used by the LRP-MIB module and LLDP-V2-LRP-EXT-MIB module, and to make those textual conventions available to future MIB modules that may find them useful. The textual conventions contained in this LRP-TC-MIB module are summarized in Table 13-2.

Table 13-2—LRP-TC-MIB structure

LRP-TC-MIB object	Reference
LrpHelloStatus	9.4.2.2
LrpAppId	9.2
LrpInetAddressInfo	C.2.2.6.2

13.2.2 Structure of the LRP-MIB

The LRP-MIB module defined in 13.5.2 has only one group of objects, as summarized in Table 13-3.

The structure of the LRP-MIB is summarized in Table 13-4.

[16]An ASCII version of this MIB module can be obtained by Web browser from the IEEE 802.1 Website at http://www.ieee802.org/1/pages/MIBS.html.

Table 13-3—LRP-MIB groups

LRP-MIB group	Required for	Reference
lrpDsDtGroup	not-controlled (Native or Proxy) systems	8.2.2, 11.4, 11.5

Table 13-4—LRP-MIB MIB structure summary

MIB Group	Table	Description
lrpDsDtGroup		
	lrpDtInstanceTable	Each entry describes one LRP-DT instance (11.4)
	lrpPortalTable	Each entry describes one Portal state machine (8.2.2)
	lrpPortalCountersTable	Each entry contains counters relating to one Portal (11.5)
	lrpDtInstanceCountersTable	Each entry contains counters relating to one LRP-DT instance (11.4)

13.2.3 Structure of the LLDP-V2-LRP-EXT-MIB

The LLDP-V2-LRP-EXT-MIB module defined in 13.5.3 is divided into three groups of objects, as summarized in Table 13-5.

Table 13-5—LLDP-V2-LRP-EXT-MIB groups

LRP-MIB group	Need	Reference
lldpV2ExtLrpControlledTcpControlGroup	Controlled systems supporting IEEE 802.1AB LLDP	6.5, C.2.2
lldpV2ExtLrpEcpTlvGroup	Any system supporting IEEE 802.1AB LLDP	6.5, C.2.1
lldpV2ExtLrpTcpTlvGroup		6.5, C.2.2

The structure of the LLDP-V2-LRP-EXT-MIB is summarized in Table 13-6.

13.3 Relationship to the LLDP-V2-TC-MIB

The LRP-MIB LLDP-V2-LRP-EXT-MIB modules defined in 13.5.2 and 13.5.3 import several textual conventions from the LLDP-V2-TC-MIB defined in IEEE Std 802.1AB.

13.4 Security considerations

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example, by using IPsec), there is no control about who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in these MIB modules.

It is recommended that implementers consider the security features as provided by the SNMPv3 framework [see IETF RFC 3410 (2002), section 8], including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Table 13-6—LLDP-V2-LRP-EXT-MIB MIB structure summary

MIB Group	Table	Description
lldpV2ExtLrpControlledTcpControlGroup		
	lldpV2ConfigLrpTcpControlledTable	Each entry provides one or two IP addresses to be advertised in LLDP LRP TCP Discovery TLVs by a Controlled system. (11.6.1.1)
lldpV2ExtLrpEcpTlvGroup		
	lldpV2ConfigLrpEcpTxTable	Each entry controls transmission of the LLDP LRP ECP Discovery TLV for one LLDP port component. (11.6.2.1)
	lldpV2LocLrpEcpTable	Each entry describes the LLDP LRP ECP Discovery TLV transmitted on one LLDP port component. (11.6.2.2)
	lldpV2RemLrpEcpTable	Each entry describes the LLDP LRP ECP Discovery TLV received on one LLDP port component for one LLDP neighbor. (11.6.2.3)
lldpV2ExtLrpTcpTlvGroup		
	lldpV2ConfigLrpTcpTxTable	Each entry controls transmission of the LLDP LRP TCP Discovery TLV for one LLDP port component. (11.6.2.4)
	lldpV2LocLrpTcpTable	Each entry describes the LLDP LRP TCP Discovery TLV transmitted on one LLDP port component. (11.6.2.5)
	lldpV2RemLrpTcpTable	Each entry describes the LLDP LRP TCP Discovery TLV received on one LLDP port component for one LLDP neighbor. (11.6.2.6)

Further, deployment of SNMP versions prior to SNMPv3 is not recommended. Instead, it is recommended to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of these MIB modules is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

13.4.1 Security considerations for the LRP-TC-MIB

This module does not define any management objects. Instead, it defines a set of textual conventions that may be used by other MIB modules to define management objects. Meaningful security considerations are contained in the following subclauses that describe security considerations for other MIB modules that define management objects with a SYNTAX of any of these textual conventions.

13.4.2 Security considerations for the LRP-MIB

There are no management objects defined in the LRP-MIB module that have a MAX-ACCESS clause of read-write.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

13.4.3 Security considerations for the LLDP-V2-LRP-EXT-MIB

There are a number of management objects defined in the LLDP-V2-LRP-EXT-MIB module that have a MAX-ACCESS clause of read-write. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

The objects that have read-write access are:

— lldpV2ConfigLrpTcpControlledTcpPortNumber
— lldpV2ConfigLrpTcpControlledIpV4Enable
— lldpV2ConfigLrpTcpControlledIpV4Address
— lldpV2ConfigLrpTcpControlledIpV6Enable
— lldpV2ConfigLrpTcpControlledIpV6Address
— lldpV2ConfigLrpEcpTxEnable
— lldpV2ConfigLrpTcpTxEnable

If any of these objects are improperly configured, whether inadvertently or with malicious intent, any or all of the following can occur. The total effect of these events depends very strongly, of course, on the specification of the LRP application running over LRP. The following events have to be understood in the context of a particular LRP application, so the ultimate effects of misconfiguration of the LRP-MIB are beyond the scope of this document. The effects include:

a) Incorrect address information can be advertised in an LRP TCP Discovery TLV, causing a neighboring system or its Proxy system to make an LRP-DT connection to the wrong Proxy system. This can make it impossible to make the LRP-DT connection, or can enable that wrong Proxy system to completely subvert the intended operation of the LRP application.
b) The transmission of LRP ECP Discovery TLVs and LRP TCP Discovery TLVs can be enabled or disabled incorrectly, causing a lack of LRP-DT connections or causing unintended LRP-DT connections to form.

13.5 MIB modules[17, 18]

13.5.1 LRP Textual conventions MIB

```
LRP-TC-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY,
    Unsigned32
        FROM SNMPv2-SMI
    ieee802dot1mibs
        FROM IEEE8021-TC-MIB
    TEXTUAL-CONVENTION
        FROM SNMPv2-TC;

ieee8021LrpTcMIB MODULE-IDENTITY
    LAST-UPDATED "202012030000Z" -- December 3, 2020
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
            "WG-URL:   http://1.ieee802.org
             WG-EMail: stds-802-1-l@ieee.org

             Contact:  IEEE 802.1 Working Group Chair
             Postal:   C/O IEEE 802.1 Working Group
                       IEEE Standards Association
                       445 Hoes Lane
                       Piscataway
                       NJ 08854
                       USA
             E-mail:   stds-802-1-chairs@ieee.org"

    DESCRIPTION
            "Textual conventions used throughout IEEE Std 802.1CS.

            Unless otherwise indicated, the references in this
            MIB module are to IEEE Std 802.1CS-2020.

            Copyright (C) IEEE (2021). This version of this MIB module
            is included in clause 13 of IEEE Std 802.1CS-2020;
            see the standard itself for full legal notices."
    REVISION    "202012030000Z" -- December 3, 2020
    DESCRIPTION "This MIB module included in IEEE Std 802.1CS-2020.
                "

   ::= { ieee802dot1mibs 34 }

--
-- ***********************************************************
-- Textual Conventions
-- ***********************************************************

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

[17]*Copyright release for MIBs*: Users of this standard may freely reproduce the MIBs contained in this subclause so that they can be used for their intended purpose.
[18]An ASCII version of each MIB module is attached to the PDF of this amendment and can also be obtained from the IEEE 802.1 Website at https://1.ieee802.org/mib-modules/.

```
        "This specifies the current state of the Hello Receive State
         Machine.  It can take the following values:

         hsLooking(1) This Portal has not yet received a successful
                      Complete Portal create request.
         hsConnecting(2) This Portal has received a successful
                      Complete Portal create request (10.2.4), and a
                      Hello LRPDU with the hsLooking status.
                      The Portal is ready to receive all LRPDUs.
         hsConnected(3) This Portal is up and ready to transfer
                      LRP application data. The Portal is allowed to
                      transmit all LRPDUs
        "
     REFERENCE
        "8.2.2.8"
     SYNTAX  INTEGER {
         hsLooking    (1),
         hsConnecting (2),
         hsConnected  (3)
     }

 LrpAppId ::= TEXTUAL-CONVENTION
     DISPLAY-HINT    "x"
     STATUS          current
     DESCRIPTION
        "Identifies an LRP application type.
         A 32 bit number. The most-significant 24 bits of the integer are
         an OUI or CID (obtainable from the IEEE Registration Authority),
         and the least-significant 8 bits are assigned by the owner of
         the OUI or CID. This creates a world-wide unique identity for
         the LRP application type.
        "
     REFERENCE    "9.2"
     SYNTAX       Unsigned32

 LrpInetAddressInfo ::= TEXTUAL-CONVENTION
     STATUS  current
     DESCRIPTION
        "An LRP TCP Discovery TLV has some number of
         Application descriptors, each of which can have either one or
         two Address info fields.  The Address info field tells whether
         the following Address field is present or not, and if present,
         whether it contains an IPv4 or an IPv6 address.
         LrpInetAddressInfo can take the following values:

         noAddress(0),   Address info present, Address field not present
         addrIPv4(1),    Address info present, Address field has IPv4
         addrIPv6(2),    Address info present, Address field has IPv6
         notPresent(256) Address info not present
        "
     REFERENCE    "C.2.2.6.2"
     SYNTAX       INTEGER {
         noAddress(0),
         addrIPv4(1),
         addrIPv6(2),
         notPresent(256)
     }

 END
```

13.5.2 LRP MIB

```
LRP-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32,
    Counter64
        FROM SNMPv2-SMI
    TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    AddressFamilyNumbers
        FROM IANA-ADDRESS-FAMILY-NUMBERS-MIB
    InetPortNumber
        FROM INET-ADDRESS-MIB
    InterfaceIndex
        FROM IF-MIB
    LldpV2ChassisIdSubtype,
    LldpV2ChassisId,
    LldpV2PortIdSubtype,
    LldpV2PortId,
    LldpV2ManAddress
        FROM LLDP-V2-TC-MIB
    ieee802dot1mibs
        FROM IEEE8021-TC-MIB
    LrpAppId
        FROM LRP-TC-MIB;

ieee8021LrpMIB MODULE-IDENTITY
    LAST-UPDATED "202012030000Z" -- December 3, 2020
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
            "WG-URL:   http://1.ieee802.org
             WG-EMail: stds-802-1-l@ieee.org

             Contact:  IEEE 802.1 Working Group Chair
             Postal:   C/O IEEE 802.1 Working Group
                       IEEE Standards Association
                       445 Hoes Lane
                       Piscataway
                       NJ 08854
                       USA
             E-mail:   stds-802-1-chairs@ieee.org"
    DESCRIPTION
           "Management Information Base module for configuration of the
            Link-local Registration Protocol.

            This MIB module supports the managed objects described in
            clause 11.

            Unless otherwise indicated, the references in this
            MIB module are to IEEE Std 802.1CS-2020.

            Copyright (C) IEEE (2021). This version of this MIB module
            is included in clause 13 of IEEE Std 802.1CS-2020;
            see the standard itself for full legal notices."

    REVISION    "202012030000Z" -- December 3, 2020
    DESCRIPTION "This MIB module included in IEEE Std 802.1CS-2020.
                "


   ::= { ieee802dot1mibs 35 }

lrpObjects               OBJECT IDENTIFIER ::= { ieee8021LrpMIB 1 }
lrpConformance           OBJECT IDENTIFIER ::= { ieee8021LrpMIB 2 }

--
-- LRP MIB Objects
--

lrpConfiguration         OBJECT IDENTIFIER ::= { lrpObjects 1 }
lrpStatistics            OBJECT IDENTIFIER ::= { lrpObjects 2 }


-- ***********************************************************
--                  L R P    C O N F I G
-- ***********************************************************


-- ******************************************************************
-- The table containing information about each LRP-DT instance.
-- ******************************************************************

lrpDtInstanceTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LrpDtInstanceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table presenting basic information about each LRP-DT instance
        in the system.
       "
    REFERENCE
       "11.4"
    ::= { lrpConfiguration 1 }

lrpDtInstanceEntry OBJECT-TYPE
    SYNTAX      LrpDtInstanceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A list of basic information about one LRP-DT instance.
       "
    INDEX { lrpDtInstNumber }
    ::= { lrpDtInstanceTable 1 }

LrpDtInstanceEntry ::= SEQUENCE {
        lrpDtInstNumber             Unsigned32,
        lrpDtInstActiveTcp          TruthValue,
        lrpDtInstAddressTypes       AddressFamilyNumbers,
        lrpDtInstMyAddress          LldpV2ManAddress,
        lrpDtInstMyTcpPort          InetPortNumber,
        lrpDtInstNeighborAddress    LldpV2ManAddress,
        lrpDtInstNeighborTcpPort    InetPortNumber
    }

lrpDtInstNumber OBJECT-TYPE
    SYNTAX      Unsigned32(1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A small integer identifying an LRP-DT instance.  Each
        LRP-DT instance in a system has a unique lrpDtInstNumber.

        This object SHALL NOT contain the value 0.
       "
    REFERENCE
       "11.2"
    ::= { lrpDtInstanceEntry 1 }

lrpDtInstActiveTcp OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "TRUE if and only if lrpDtInstAddressTypes indicates an IPv4
        or IPv6 address, and this LRP-DT instance uses an active TCP
        OPEN, as opposed to a passive TCP OPEN, to initiate the TCP
        connection.
       "
    REFERENCE
       "11.4, 7.3.2.1"
    ::= { lrpDtInstanceEntry 2 }

lrpDtInstAddressTypes OBJECT-TYPE
    SYNTAX      AddressFamilyNumbers
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "An enumerated value specifying the format of the addresses
        in lrpDtInstMyAddress and lrpDtInstNeighborAddress.

        If lrpDtInstAddressTypes has a value indicating a type of
        Inetnet Protocol address, then this LRP-DT instance uses TCP.
        Otherwise, it uses ECP.
       "
    REFERENCE
       "11.4, 7.3.2.2, 7.3.2.4"
    ::= { lrpDtInstanceEntry 3 }

lrpDtInstMyAddress OBJECT-TYPE
    SYNTAX      LldpV2ManAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The address used by the local end of the LRP-DT instance.  The
        format of the address is determined by lrpDtInstAddressTypes.
       "
    REFERENCE
       "11.4, 7.3.2.2"
    ::= { lrpDtInstanceEntry 4 }

lrpDtInstMyTcpPort OBJECT-TYPE
```

```
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The local TCP port number used for the TCP connection, or 0,
        if this connection uses ECP, instead of TCP.
       "
    REFERENCE
       "11.4, 7.3.2.3"
    ::= { lrpDtInstanceEntry 5 }

lrpDtInstNeighborAddress OBJECT-TYPE
    SYNTAX      LldpV2ManAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The address used by the partner end of the LRP-DT instance, if
        lrpDtInstActiveTcp is TRUE or lrpDtInstAddressTypes indicates
        a MAC address.  Otherwise (this LRP-DT instance was created
        from a passive TCP OPEN), lrpDtInstNeighborAddress contains a
        zero-length string.  The format of the address is determined by
        lrpDtInstAddressTypes.
       "
    REFERENCE
       "11.4, 7.3.2.4"
    ::= { lrpDtInstanceEntry 6 }

lrpDtInstNeighborTcpPort OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The remote TCP port number used for the TCP connection, or 0,
        if this connection uses ECP, instead of TCP.
       "
    REFERENCE
       "11.4, 7.3.2.5"
    ::= { lrpDtInstanceEntry 7 }


-- ******************************************************************
-- The table containing information about each LRP-DS Portal.
-- ******************************************************************

lrpPortalTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LrpPortalEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table containing the per-portal set of counters that record
        LRP events.  There is an entry in the table for every portal in
        a system.
       "
    REFERENCE
       "8.2.2"
    ::= { lrpConfiguration 2 }

lrpPortalEntry OBJECT-TYPE
    SYNTAX      LrpPortalEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A list of counters for events occurring on one Portal.
       "
    INDEX { lrpPortalNumber }
    ::= { lrpPortalTable 1 }

LrpPortalEntry ::= SEQUENCE {
        lrpPortalNumber              Unsigned32,
        lrpPortalIfIndex             InterfaceIndex,
        lrpPortalDtInstanceIndex     Unsigned32,
        lrpPortalAppId               LrpAppId,
        lrpPortalMyChassisIdType     LldpV2ChassisIdSubtype,
        lrpPortalMyChassisId         LldpV2ChassisId,
        lrpPortalMyPortIdType        LldpV2PortIdSubtype,
        lrpPortalMyPortId            LldpV2PortId,
        lrpPortalNbrChassisIdType    LldpV2ChassisIdSubtype,
        lrpPortalNbrChassisId        LldpV2ChassisId,
        lrpPortalNbrPortIdType       LldpV2PortIdSubtype,
        lrpPortalNbrPortId           LldpV2PortId,
        lrpPortalLocalOverflow       TruthValue
    }

lrpPortalNumber OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A small integer identifying a portal.  Each portal in a system
        has a unique lrpPortalNumber.

        This object SHALL NOT contain the value 0.
       "
    ::= { lrpPortalEntry 1 }

lrpPortalIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The interface index identifying the target port to which this
        portal is attached.  The value is 0, if there is none.
       "
    ::= { lrpPortalEntry 2 }

lrpPortalDtInstanceIndex OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The same value as the lrpDtInstNumber object of the
        lrpDtInstanceEntry describing the LRP-DT instance to which this
        Portal is attached.
       "
    REFERENCE
       "8.2.2.1"
    ::= { lrpPortalEntry 3 }
```

```
lrpPortalAppId OBJECT-TYPE
    SYNTAX      LrpAppId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The application ID used for this Portal.
       "
    REFERENCE
       "8.2.2.1"
    ::= { lrpPortalEntry 4 }

lrpPortalMyChassisIdType OBJECT-TYPE
    SYNTAX      LldpV2ChassisIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The My Chassis ID TLV type field used for this Portal.
       "
    REFERENCE
       "8.2.2.2"
    ::= { lrpPortalEntry 5 }

lrpPortalMyChassisId OBJECT-TYPE
    SYNTAX      LldpV2ChassisId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The My Chassis ID TLV Chassis ID field used for this Portal.
       "
    REFERENCE
       "8.2.2.2"
    ::= { lrpPortalEntry 6 }

lrpPortalMyPortIdType OBJECT-TYPE
    SYNTAX      LldpV2PortIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The My Port ID TLV type field used for this Portal.
       "
    REFERENCE
       "8.2.2.3"
    ::= { lrpPortalEntry 7 }

lrpPortalMyPortId OBJECT-TYPE
    SYNTAX      LldpV2PortId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The My Port ID TLV Port ID field used for this Portal.
       "
    REFERENCE
       "8.2.2.3"
    ::= { lrpPortalEntry 8 }

lrpPortalNbrChassisIdType OBJECT-TYPE
    SYNTAX      LldpV2ChassisIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Neighbor Chassis ID TLV type field used for this Portal.
       "
    REFERENCE
       "8.2.2.2"
    ::= { lrpPortalEntry 9 }

lrpPortalNbrChassisId OBJECT-TYPE
    SYNTAX      LldpV2ChassisId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Neighbor Chassis ID TLV Chassis ID field used for this
        Portal.
       "
    REFERENCE
       "8.2.2.2"
    ::= { lrpPortalEntry 10 }

lrpPortalNbrPortIdType OBJECT-TYPE
    SYNTAX      LldpV2PortIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Neighbor Port ID TLV type field used for this Portal.
       "
    REFERENCE
       "8.2.2.3"
    ::= { lrpPortalEntry 11 }

lrpPortalNbrPortId OBJECT-TYPE
    SYNTAX      LldpV2PortId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Neighbor Port ID TLV Port ID field used for this Portal.
       "
    REFERENCE
       "8.2.2.3"
    ::= { lrpPortalEntry 12 }

lrpPortalLocalOverflow OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "A Boolean indicating whether or not the local registrar
        database has overflowed its alloted memory.
       "
    REFERENCE
       "8.2.2.10"
    ::= { lrpPortalEntry 13 }

-- ************************************************************
--          P O R T A L    S T A T I S T I C S
-- ************************************************************

lrpPortalCountersTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LrpPortalCountersEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table containing the per-portal set of counters that record
        LRP events.  There is an entry in the table for every portal in
        a system.
       "
    REFERENCE
       "11.5"
    ::= { lrpStatistics 1 }

lrpPortalCountersEntry OBJECT-TYPE
    SYNTAX      LrpPortalCountersEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A list of counters for events occurring on one Portal.
       "
    INDEX { lrpPortalNumber }
    ::= { lrpPortalCountersTable 1 }

LrpPortalCountersEntry ::= SEQUENCE {
        lrpPortalApplicantActiveRecords Unsigned32,
        lrpPptCtRegistrarActiveRecords  Unsigned32,
        lrpPptCtSentHellos              Counter64,
        lrpPptCtAcceptedHellos          Counter64,
        lrpPptCtDiscardedHellos         Counter64,
        lrpPptCtSentRecords             Counter64,
        lrpPptCtAcceptedRecords         Counter64,
        lrpPptCtDiscardedRecords        Counter64,
        lrpPptCtRecordErrors            Counter64,
        lrpPptCtSentPartials            Counter64,
        lrpPptCtAcceptedPartials        Counter64,
        lrpPptCtDiscardedPartials       Counter64,
        lrpPptCtSentCompletes           Counter64,
        lrpPptCtAcceptedCompletes       Counter64,
        lrpPptCtDiscardedCompletes      Counter64,
        lrpPptCtDiscardedUnknowns       Counter64
    }

lrpPortalApplicantActiveRecords OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The number of records in the applicant database.
       "
    REFERENCE
       "11.5.1"
    ::= { lrpPortalCountersEntry 1 }

lrpPptCtRegistrarActiveRecords OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The number of records in the registrar database.
       "
    REFERENCE
```

```
           "11.5.2"
        ::= { lrpPortalCountersEntry 2 }

    lrpPptCtSentHellos OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "Incremented once for each Hello LRPDU transmitted by the
            Send Hello state machines.
           "
        REFERENCE
           "11.5.3"
        ::= { lrpPortalCountersEntry 3 }

    lrpPptCtAcceptedHellos OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "Incremented once for each Hello LRPDU received by the
            Receive Hello state machine.
           "
        REFERENCE
           "11.5.4"
        ::= { lrpPortalCountersEntry 4 }

    lrpPptCtDiscardedHellos OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "Incremented once for each invalid Hello LRPDU discarded by the
            Receive Hello state machine
           "
        REFERENCE
           "11.5.5"
        ::= { lrpPortalCountersEntry 5 }

    lrpPptCtSentRecords OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "Incremented once for each Record LRPDU transmitted by the
            Applicant state machine.
           "
        REFERENCE
           "11.5.6"
        ::= { lrpPortalCountersEntry 6 }

    lrpPptCtAcceptedRecords OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "Incremented once for each valid Record LRPDU received by the
            Partial list state machine.
           "
    REFERENCE
       "11.5.7"
    ::= { lrpPortalCountersEntry 7 }

lrpPptCtDiscardedRecords OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each invalid Record LRPDU discarded by the
        Partial list state machine.
       "
    REFERENCE
       "11.5.8"
    ::= { lrpPortalCountersEntry 8 }

lrpPptCtRecordErrors OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each record discarded from a Record LRPDU
        because of inconsistencies among the Checksum, Application data,
        and Data length fields.
       "
    REFERENCE
       "11.5.9"
    ::= { lrpPortalCountersEntry 9 }

lrpPptCtSentPartials OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each Partial List LRPDU transmitted by the
        Applicant state machine.
       "
    REFERENCE
       "11.5.10"
    ::= { lrpPortalCountersEntry 10 }

lrpPptCtAcceptedPartials OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each valid Partial List LRPDU received by
        the Applicant state machine.
       "
    REFERENCE
       "11.5.11"
    ::= { lrpPortalCountersEntry 11 }

lrpPptCtDiscardedPartials OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each invalid Partial List LRPDU discarded
```

```
        by the Applicant state machine.
       "
    REFERENCE
       "11.5.12"
    ::= { lrpPortalCountersEntry 12 }

lrpPptCtSentCompletes OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each Complete List LRPDU transmitted by
        the Applicant state machine.
       "
    REFERENCE
       "11.5.13"
    ::= { lrpPortalCountersEntry 13 }

lrpPptCtAcceptedCompletes OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each valid Complete List LRPDU received
        by the Applicant state machine.
       "
    REFERENCE
       "11.5.14"
    ::= { lrpPortalCountersEntry 14 }

lrpPptCtDiscardedCompletes OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each invalid Complete List LRPDU discarded
        by the Applicant state machine.
       "
    REFERENCE
       "11.5.15"
    ::= { lrpPortalCountersEntry 15 }

lrpPptCtDiscardedUnknowns OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Incremented once for each LRPDU of unknown type discarded by
        the Applicant state machine or Partial list state machine.
       "
    REFERENCE
       "11.5.16"
    ::= { lrpPortalCountersEntry 16 }


-- ******************************************************************
--         L R P - D T   I N S T A N C E   S T A T I S T I C S
-- ******************************************************************
```

```
lrpDtInstanceCountersTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LrpDtInstanceCountersEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table containing the per-LRP-DT instance set of counters that
        record LRP events.  There is an entry in the table for every
        LRP-DT instance in a system.
       "
    REFERENCE
       "11.4"
    ::= { lrpStatistics 2 }

lrpDtInstanceCountersEntry OBJECT-TYPE
    SYNTAX      LrpDtInstanceCountersEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A list of statistics about one LRP-DT instance.
       "
    INDEX { lrpDtInstNumber }
    ::= { lrpDtInstanceCountersTable 1 }

LrpDtInstanceCountersEntry ::= SEQUENCE {
        lrpDtInstDiscardedLrpdus             Counter64
    }

lrpDtInstDiscardedLrpdus OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The number of received Link-local Registration Protocol Data
        Units discarded by the LRP-DT instance because it could not
        determine to which Portal it should be given.
       "
    REFERENCE
       "11.4, 11.4.1"
    ::= { lrpDtInstanceCountersEntry 1 }


--
-- ***********************************************************
--          L R P   M I B   C O N F O R M A N C E
-- ***********************************************************
--

lrpCompliances OBJECT IDENTIFIER ::= { lrpConformance 1 }
lrpGroups      OBJECT IDENTIFIER ::= { lrpConformance 2 }

-- compliance statements

lrpCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
           "A compliance statement for all SNMP entities that
            implement the LRP MIB.

            This version defines compliance requirements for
```

```
             LRP MIB module.
            "
    MODULE  -- this module
        MANDATORY-GROUPS {
            lrpDsDtGroup
        }

    ::= { lrpCompliances 1 }

-- MIB groupings

lrpDsDtGroup    OBJECT-GROUP
    OBJECTS {
        lrpDtInstActiveTcp,
        lrpDtInstAddressTypes,
        lrpDtInstMyAddress,
        lrpDtInstMyTcpPort,
        lrpDtInstNeighborAddress,
        lrpDtInstNeighborTcpPort,
        lrpPortalIfIndex,
        lrpPortalDtInstanceIndex,
        lrpPortalAppId,
        lrpPortalMyChassisIdType,
        lrpPortalMyChassisId,
        lrpPortalMyPortIdType,
        lrpPortalMyPortId,
        lrpPortalNbrChassisIdType,
        lrpPortalNbrChassisId,
        lrpPortalNbrPortIdType,
        lrpPortalNbrPortId,
        lrpPortalLocalOverflow,
        lrpPortalApplicantActiveRecords,
        lrpPptCtRegistrarActiveRecords,
        lrpPptCtSentHellos,
        lrpPptCtAcceptedHellos,
        lrpPptCtDiscardedHellos,
        lrpPptCtSentRecords,
        lrpPptCtAcceptedRecords,
        lrpPptCtDiscardedRecords,
        lrpPptCtRecordErrors,
        lrpPptCtSentPartials,
        lrpPptCtAcceptedPartials,
        lrpPptCtDiscardedPartials,
        lrpPptCtSentCompletes,
        lrpPptCtAcceptedCompletes,
        lrpPptCtDiscardedCompletes,
        lrpPptCtDiscardedUnknowns,
        lrpDtInstDiscardedLrpdus
    }
    STATUS  current
    DESCRIPTION
           "The collection of objects which are used to monitor the
            status of LRP-DS and LRP-DT.
           "
    ::= { lrpGroups 1 }

END
```

13.5.3 LLDPv2 LRP extension MIB

```
LLDP-V2-LRP-EXT-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32
        FROM SNMPv2-SMI
    TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    TimeFilter
        FROM RMON2-MIB
    InterfaceIndex
        FROM IF-MIB
    InetAddress,
    InetAddressIPv4,
    InetAddressIPv6,
    InetPortNumber
        FROM INET-ADDRESS-MIB
    LldpV2DestAddressTableIndex
        FROM LLDP-V2-TC-MIB
    lldpXdot1StandAloneExtensions
        FROM LLDP-EXT-DOT1-EVB-EXTENSIONS-MIB
    LrpAppId,
    LrpInetAddressInfo
        FROM LRP-TC-MIB;

lldpXDot1LrpExtensions MODULE-IDENTITY
    LAST-UPDATED "202012030000Z" -- December 3, 2020
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
       "WG-URL: http://www.ieee802.org/1/
        WG-EMail: stds-802-1-l@ieee.org

        Contact: IEEE 802.1 Working Group Chair
        Postal:  C/O IEEE 802.1 Working Group
                 IEEE Standards Association
                 445 Hoes Lane
                 Piscataway
                 NJ 08854
                 USA
        E-mail:  stds-802-1-chairs@ieee.org"
    DESCRIPTION
       "The LLDP Management Information Base extension module for IEEE
        802.1 organizationally-defined discovery information, as
        specified in IEEE Std 802.1CS, Link-local Registration Protocol
        (LRP).

        The Link-Layer Discovery Protocol (LLDP) is defined in
        IEEE Std 802.1AB.

              lldpXdot1StandAloneExtensions  is  the  OUI  for  LLDP-EXT-DOT1-EVB-
EXTENSIONS-MIB.
        which defines managed objects for IEEE 802.1-defined
        organizationally-specified LLDP Type-Length Value (TLV)
        discovery information. lldpXdot1StandAloneExtensions is branched
        from lldpV2Extensions (defined in LLDP-V2-MIB) using the
        Organizationally Unique Identifier (OUI) value 00-80-C1, which
        belongs to IEEE 802.1. An OUI is a 24 bit globally-unique number
        assigned by the IEEE Registration Authority -- see:

            http://standards.ieee.org/develop/regauth/oui/index.html

        In turn, lldpXDot1LrpExtensions and lldpV2ExtLrpConformance are
        branched from lldpXdot1StandAloneExtensions, and thus are also
        extensions from the IEEE 802.1 OUI.

        Unless otherwise indicated, the references in this MIB module
        are to IEEE Std 802.1CS-2020.

        Copyright (C) IEEE (2020). This version of this MIB module is
        included in clause 13 of IEEE Std 802.1CS-2020; see the
        standard itself for full legal notices."

    REVISION "202012030000Z" -- December 3, 2020
    DESCRIPTION
       "This MIB module included in IEEE Std 802.1CS-2020.
       "

   ::= { lldpXdot1StandAloneExtensions 3 }

------------------------------------------------------------------------
------------------------------------------------------------------------
--
-- Organizationally Defined Information Extension - IEEE 802.1
-- Definitions to support the IEEE Std 802.1AB LLDP TLVs defined in
-- IEEE Std 802.1CS Link-local Registration Protocol (LRP)
--
------------------------------------------------------------------------
------------------------------------------------------------------------

lldpV2ExtLrpObjects    OBJECT IDENTIFIER ::= { lldpXDot1LrpExtensions 1 }

-- LLDP IEEE 802.1CS extension MIB groups

lldpV2ExtConfigLrp     OBJECT IDENTIFIER ::= { lldpV2ExtLrpObjects 1 }
lldpV2ExtLrpLocalData  OBJECT IDENTIFIER ::= { lldpV2ExtLrpObjects 2 }
lldpV2ExtLrpRemoteData OBJECT IDENTIFIER ::= { lldpV2ExtLrpObjects 3 }

------------------------------------------------------------------------
-- IEEE 802.1 - Configuration for the LRP TLV set
------------------------------------------------------------------------

--
-- The table specifying, for each LRP application, what IP
-- addresses to advertise in LRP TCP Discovery TLVs in a
-- Controlled system.
--

lldpV2ConfigLrpTcpControlledTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2LrpConfigTcpControlledEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table specifying what IP addresses are to be advertised as
        the address of the Proxy system controlling this Controlled
        system, for each particular LRP application. These IP addresses
        and application identifiers can be transmitted in
        LRP TCP Discovery TLVs.
       "
    REFERENCE
       "11.6.1.1"
    ::= { lldpV2ExtConfigLrp 1 }

lldpV2ConfigLrpTcpControlledEntry OBJECT-TYPE
    SYNTAX      LldpV2LrpConfigTcpControlledEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table specifying what IP addresses are to be advertised as
        the address of the Proxy system controlling this Controlled
        system, for a particular LRP application. These IP addresses
        and application identifiers can be transmitted in
        LRP TCP Discovery TLVs.
       "
    INDEX { lldpV2ConfigLrpTcpControlledApplicationId }
    ::= { lldpV2ConfigLrpTcpControlledTable 1 }

LldpV2LrpConfigTcpControlledEntry ::= SEQUENCE {
        lldpV2ConfigLrpTcpControlledApplicationId   LrpAppId,
        lldpV2ConfigLrpTcpControlledTcpPortNumber   InetPortNumber,
        lldpV2ConfigLrpTcpControlledIpV4Enable      TruthValue,
        lldpV2ConfigLrpTcpControlledIpV4Address     InetAddressIPv4,
        lldpV2ConfigLrpTcpControlledIpV6Enable      TruthValue,
        lldpV2ConfigLrpTcpControlledIpV6Address     InetAddressIPv6

    }

lldpV2ConfigLrpTcpControlledApplicationId OBJECT-TYPE
    SYNTAX      LrpAppId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The application identifier to which the rest of the
        lldpV2ConfigLrpTcpControlledEntry applies.
       "
    REFERENCE
       "9.2"
    ::= { lldpV2ConfigLrpTcpControlledEntry 1 }

lldpV2ConfigLrpTcpControlledTcpPortNumber OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "The destination TCP Port number to which TCP connections for
        LRP to the addresses in lldpV2ConfigLrpTcpControlledIpV4Address
        or lldpV2ConfigLrpTcpControlledIpV6Address, for the
        application in lldpV2ConfigLrpTcpControlledApplicationId, are
        to be made.

        If this object contains the value 0, then no
```

```
            Application descriptor with the indexed application ID is
            transmitted.

            The value of this object is restored from non-volatile
            storage after a re-initialization of the management system.
           "
       REFERENCE
           "C.2.2.6.1"
       ::= { lldpV2ConfigLrpTcpControlledEntry 2 }

   lldpV2ConfigLrpTcpControlledIpV4Enable OBJECT-TYPE
       SYNTAX      TruthValue
       MAX-ACCESS  read-write
       STATUS      current
       DESCRIPTION
           "Specifies whether or not the indexed LRP application is
            available through the LRP-DT TCP mechanism using TCP over IPv4.
            It thus controls whether the LRP TCP Discovery TLVs transmitted
            from this Controlled system include the IPv4 address in
            lldpV2ConfigLrpTcpControlledIpV4Address in an
            Application descriptor containing the indexed application ID.

            If lldpV2ConfigLrpTcpControlledIpV4Enable and
            lldpV2ConfigLrpTcpControlledIpV4Enable are both false(2), then
            no Application descriptor with the indexed application ID is
            transmitted.

            The value of this object is restored from non-volatile
            storage after a re-initialization of the management system.
           "
       REFERENCE
           "C.2.2.6.2"
       ::= { lldpV2ConfigLrpTcpControlledEntry 3 }

   lldpV2ConfigLrpTcpControlledIpV4Address OBJECT-TYPE
       SYNTAX      InetAddressIPv4
       MAX-ACCESS  read-write
       STATUS      current
       DESCRIPTION
           "Specifies an IPv4 address to be advertised in all of the
            LRP TCP Discovery TLVs that carry the indexed application ID
            that are transmitted by this Controlled system.

            The value of this object is restored from non-volatile
            storage after a re-initialization of the management system.
           "
       REFERENCE
           "C.2.2.6.3"
       ::= { lldpV2ConfigLrpTcpControlledEntry 4 }

   lldpV2ConfigLrpTcpControlledIpV6Enable OBJECT-TYPE
       SYNTAX      TruthValue
       MAX-ACCESS  read-write
       STATUS      current
       DESCRIPTION
           "Specifies whether or not the indexed LRP application is
            available through the LRP-DT TCP mechanism using TCP over IPv6.
            It thus controls whether the LRP TCP Discovery TLVs transmitted
            from this Controlled system include the IPv6 address in
```

```
         lldpV2ConfigLrpTcpControlledIpV6Address in an
         Application descriptor containing the indexed application ID.

         If lldpV2ConfigLrpTcpControlledIpV4Enable and
         lldpV2ConfigLrpTcpControlledIpV4Enable are both false(2), then
         no Application descriptor with the indexed application ID is
         transmitted.

         The value of this object is restored from non-volatile
         storage after a re-initialization of the management system.
        "
    REFERENCE
        "C.2.2.6.2"
    ::= { lldpV2ConfigLrpTcpControlledEntry 5 }

lldpV2ConfigLrpTcpControlledIpV6Address OBJECT-TYPE
    SYNTAX      InetAddressIPv6
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "Specifies an IPv6 address to be advertised in all of the
         LRP TCP Discovery TLVs that carry the indexed application ID
         that are transmitted by this Controlled system.

         The value of this object is restored from non-volatile
         storage after a re-initialization of the management system.
        "
    REFERENCE
        "C.2.2.6.3"
    ::= { lldpV2ConfigLrpTcpControlledEntry 6 }

--
-- lldpV2ConfigLrpEcpTxTable: configure the transmission of the
--                 LRP ECP Discovery TLVs on a set of ports.
--

lldpV2ConfigLrpEcpTxTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2ConfigLrpEcpTxEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one or more rows per physical network
         connection known to this agent.  The agent may wish to
         ensure that only one lldpV2ConfigLrpEcpTxEntry is present for
         each local port, or it may choose to maintain multiple
         entries for the same local port."
    REFERENCE
        "11.6.2.1"
    ::= { lldpV2ExtLrpLocalData 1 }

lldpV2ConfigLrpEcpTxEntry OBJECT-TYPE
    SYNTAX      LldpV2ConfigLrpEcpTxEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Information about a particular port component."
    INDEX   { lldpV2ConfigLrpEcpTxLocalIfIndex,
              lldpV2ConfigLrpEcpTxLocalDestMACAddress }
    ::= { lldpV2ConfigLrpEcpTxTable 1 }
```

```
LldpV2ConfigLrpEcpTxEntry ::= SEQUENCE {
    lldpV2ConfigLrpEcpTxLocalIfIndex        InterfaceIndex,
    lldpV2ConfigLrpEcpTxLocalDestMACAddress LldpV2DestAddressTableIndex,
    lldpV2ConfigLrpEcpTxEnable              TruthValue
    }

lldpV2ConfigLrpEcpTxLocalIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The interface index value used to identify the port
        associated with this entry. Its value is an index
        into the interfaces MIB

        The value of this object is used as an index to the
        lldpV2ConfigLrpEcpTxTable.
       "
    ::= { lldpV2ConfigLrpEcpTxEntry 1 }

lldpV2ConfigLrpEcpTxLocalDestMACAddress OBJECT-TYPE
    SYNTAX      LldpV2DestAddressTableIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The index value used to identify the LLDPDU frame destination
        MAC address associated with this entry. Its value identifies
        the row in the lldpV2DestAddressTable where the MAC address
        can be found.

        The value of this object is used as an index to the
        lldpV2ConfigLrpEcpTxTable.
       "
    ::= { lldpV2ConfigLrpEcpTxEntry 2 }

lldpV2ConfigLrpEcpTxEnable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "The lldpV2ConfigLrpEcpTxEnable, which is defined
        as a truth value and configured by the network management,
        determines whether the IEEE 802.1 organizationally defined
        LRP ECP Discovery TLV transmission is allowed on a given
        LLDP transmission-capable port component.

        The value of this object is restored from non-volatile
        storage after a re-initialization of the management system."
    DEFVAL  { false }
    ::= { lldpV2ConfigLrpEcpTxEntry 3 }


--
-- lldpV2ConfigLrpTcpTxTable: configure the transmission of the
--                 LRP TCP Discovery TLVs on a set of ports.
--

lldpV2ConfigLrpTcpTxTable OBJECT-TYPE
```

```
    SYNTAX      SEQUENCE OF LldpV2ConfigLrpTcpTxEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This table contains one or more rows per physical network
        connection known to this agent.  The agent may wish to
        ensure that only one lldpV2ConfigLrpTcpTxEntry is present for
        each local port, or it may choose to maintain multiple
        entries for the same local port."
    ::= { lldpV2ExtLrpLocalData 2 }

lldpV2ConfigLrpTcpTxEntry OBJECT-TYPE
    SYNTAX      LldpV2ConfigLrpTcpTxEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Information about a particular port component."
    INDEX   { lldpV2ConfigLrpTcpTxLocalIfIndex,
              lldpV2ConfigLrpTcpTxLocalDestMACAddress }
    ::= { lldpV2ConfigLrpTcpTxTable 1 }

LldpV2ConfigLrpTcpTxEntry ::= SEQUENCE {
    lldpV2ConfigLrpTcpTxLocalIfIndex        InterfaceIndex,
    lldpV2ConfigLrpTcpTxLocalDestMACAddress LldpV2DestAddressTableIndex,
    lldpV2ConfigLrpTcpTxEnable              TruthValue
    }

lldpV2ConfigLrpTcpTxLocalIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The interface index value used to identify the port
        associated with this entry. Its value is an index
        into the interfaces MIB

        The value of this object is used as an index to the
        lldpV2ConfigLrpTcpTxTable.
       "
    ::= { lldpV2ConfigLrpTcpTxEntry 1 }

lldpV2ConfigLrpTcpTxLocalDestMACAddress OBJECT-TYPE
    SYNTAX      LldpV2DestAddressTableIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The index value used to identify the LLDPDU frame destination
        MAC address associated with this entry. Its value identifies
        the row in the lldpV2DestAddressTable where the MAC address
        can be found.

        The value of this object is used as an index to the
        lldpV2ConfigLrpTcpTxTable.
       "
    ::= { lldpV2ConfigLrpTcpTxEntry 2 }

lldpV2ConfigLrpTcpTxEnable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
           "The lldpV2ConfigLrpTcpTxEnable, which is defined
            as a truth value and configured by the network management,
            determines whether the IEEE 802.1 organizationally defined
            LRP TCP Discovery TLV transmission is allowed on a given
            LLDP transmission-capable port component.

            The value of this object is restored from non-volatile
            storage after a re-initialization of the management system."
    REFERENCE
            "9.1.2.1 of IEEE Std 802.1AB-2016"
    DEFVAL  { false }
    ::= { lldpV2ConfigLrpTcpTxEntry 3 }

------------------------------------------------------------------------
-- IEEE 802.1CS LRP LLDP TLVs - Local System Information
------------------------------------------------------------------------

--
-- lldpV2LocLrpEcpTable
--

lldpV2LocLrpEcpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2LocLrpEcpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This table contains one or more rows per physical network
        connection known to this agent.  The agent may wish to
        ensure that only one lldpV2ExtLrpLocEntry is present for
        each local port, or it may choose to maintain multiple
        lldpV2ExtLrpLocEntries for the same local port.
       "
    REFERENCE
       "11.6.2.2"
    ::= { lldpV2ExtLrpLocalData 3 }

lldpV2LocLrpEcpEntry OBJECT-TYPE
    SYNTAX      LldpV2LocLrpEcpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Information about the C.2.1LRP ECP Discovery TLV that can
        be transmitted from a particular LLDP port component.
        Note that this MIB supports the transmission of only one
        LRP ECP Discovery TLV per port component.
       "
    INDEX   { lldpV2LocLrpEcpLocalIfIndex,
              lldpV2LocLrpEcpLocalDestMACAddress,
              lldpV2LocLrpEcpApplicationIndex }
    ::= { lldpV2LocLrpEcpTable 1 }

LldpV2LocLrpEcpEntry ::= SEQUENCE {
    lldpV2LocLrpEcpLocalIfIndex         InterfaceIndex,
    lldpV2LocLrpEcpLocalDestMACAddress  LldpV2DestAddressTableIndex,
    lldpV2LocLrpEcpApplicationIndex     Unsigned32,
    lldpV2LocLrpEcpApplicationId        LrpAppId
    }
```

```
lldpV2LocLrpEcpLocalIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The interface index value used to identify the port
        associated with this entry. Its value is an index
        into the interfaces MIB

        The value of this object is used as an index to the
        lldpV2LocLrpEcpTable.
       "
    ::= { lldpV2LocLrpEcpEntry 1 }

lldpV2LocLrpEcpLocalDestMACAddress OBJECT-TYPE
    SYNTAX      LldpV2DestAddressTableIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The index value used to identify the LLDPDU frame destination
        MAC address associated with this entry. Its value identifies
        the row in the lldpV2DestAddressTable where the MAC address
        can be found.

        The value of this object is used as an index to the
        lldpV2LocLrpEcpTable.
       "
    ::= { lldpV2LocLrpEcpEntry 2 }

lldpV2LocLrpEcpApplicationIndex OBJECT-TYPE
    SYNTAX      Unsigned32 (0..255)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A small integer that selects one entry in the
        lldpV2LocLrpEcpTable.  For every entry in lldpV2LocLrpEcpEntry,
        there is one Application descriptor in the transmitted
        LRP ECP Discovery TLV.

        The value of the transmitted Application count field in the
        LRP ECP Discovery TLV is equal to the number of different values
        of lldpV2LocLrpEcpApplicationIndex for this port component.
       "
    REFERENCE "C.2.1.5, C.2.1.6"
    ::= { lldpV2LocLrpEcpEntry 3 }

lldpV2LocLrpEcpApplicationId OBJECT-TYPE
    SYNTAX      LrpAppId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The AppId in one Application descriptor in the transmitted
        LRP ECP Discovery TLV.
       "
    REFERENCE "C.2.1.5, C.2.1.6"
    ::= { lldpV2LocLrpEcpEntry 4 }

--
```