ANSI/IEEE Std 802.1E, 1994 Edition

(Incorporates ANSI/IEEE Std 802.1E-1990 and
ANSI/IEEE Std 802.1m-1993)

(Adopted by ISO/IEC and redesignated as
ISO/IEC 15802-4:1994)

IEEE Standard for Information technology—

Telecommunications and information exchange between systems—

Local and metropolitan area networks—

Common specifications

Part 4: System load protocol

Adopted by the ISO/IEC and redesignated as ISO/IEC 15802-4:1994

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Abstract: Services and protocol elements that permit the downloading of memory images to data processing equipment attached to IEEE 802 Local and Metropolitan Area Networks are defined. The protocol makes use of the group addressing capabilities inherent in LAN/MAN technologies to permit simultaneous loading of the same memory image to multiple destination systems. The standard includes the specification of managed objects that permit the operation of the load protocol to be remotely managed.
Keywords: local area networks, management; metropolitan area networks, management; system load protocol

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 1-55937-341-5

May 16, 1994 *SH16535*

ANSI/IEEE Std 802.1E, 1994 Edition

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

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. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

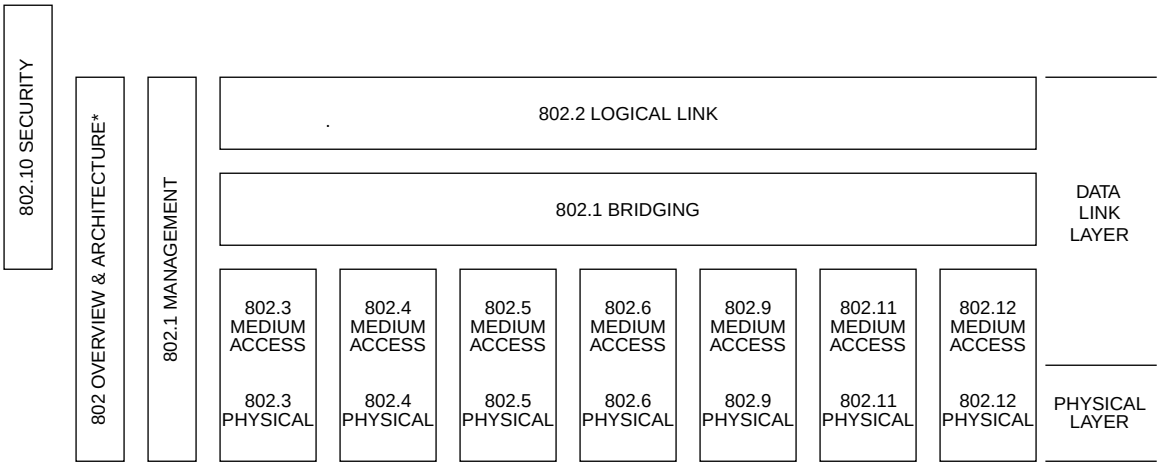
Comments on standards and requests for interpretations should be addressed to:

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

Introduction

(This introduction is not a part of ANSI/IEEE Std 802.1E, 1994 Edition or of ISO/IEC 15802-4 : 1994.)

This standard is part of a family of standards for local and metropolitan area networks. The relationship between the standard and other members of the family is shown below. (The numbers in the figure refer to IEEE standard numbers.)



* Formerly IEEE Std 802.1A.

This family of standards deals with the Physical and Data Link layers as defined by the International Organization for Standardization (ISO) Open Systems Interconnection Basic Reference Model (ISO 7498 : 1984). The access standards define several types of medium access technologies and associated physical media, each appropriate for particular applications or system objectives. Other types are under investigation.

The standards defining these technologies are as follows:

- IEEE Std 802[†]: Overview and Architecture. This standard provides an overview to the family of IEEE 802 Standards. This document forms part of the 802.1 scope of work.
- ISO/IEC DIS 15802-2 [IEEE Stds 802.1B and 802.1k]: LAN/MAN Management. Defines an Open Systems Interconnection (OSI) management-compatible architecture, and services and protocol elements for use in a LAN/MAN environment for performing remote management.
- ISO/IEC 10038 [ANSI/IEEE Std 802.1D]: MAC Bridging. Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.
- ISO/IEC 15802-4 [IEEE Std 802.1E]: System Load Protocol. Specifies a set of services and protocol for those aspects of management concerned with the loading of systems on IEEE 802 LANs.
- ISO 8802-2 [ANSI/IEEE Std 802.2]: Logical Link Control

[†]The 802 Architecture and Overview Specification, originally known as IEEE Std 802.1A, has been renumbered as IEEE Std 802. This has been done to accommodate recognition of the base standard in a family of standards. References to IEEE Std 802.1A should be considered as references to IEEE Std 802.

- ISO/IEC 8802-3 [ANSI/IEEE Std 802.3]: CSMA/CD Access Method and Physical Layer Specifications
- ISO/IEC 8802-4 [ANSI/IEEE Std 802.4]: Token Bus Access Method and Physical Layer Specifications
- ISO/IEC 8802-5 [ANSI/IEEE Std 802.5]: Token Ring Access Method and Physical Layer Specifications
- ISO/IEC 8802-6 [ANSI/IEEE Std 802.6]: Distributed Queue Dual Bus Access Method/Physical Layer Specifications
- IEEE Std 802.10: Interoperable LAN/MAN Security, *Currently Contains* Secure Data Exchange

In addition to the family of standards, the following is a recommended practice for a common technology:

- IEEE Std 802.7: IEEE Recommended Practice for Broadband Local Area Networks

The following additional working groups have authorized standards projects under development:

- IEEE 802.9 Integrated Services (IS) LAN Interface at the Medium Access Control (MAC) and Physical Layers
- IEEE 802.11 Wireless LAN Medium Access Control (MAC)/Physical Layer Specifications
- IEEE 802.12 Demand Priority Access Method and Physical Layer Specifications

The reader of this standard is urged to become familiar with the complete family of standards.

Conformance test methodology

An additional standards series, identified by the number 1802, has been established to identify the conformance test methodology documents for the 802 family of standards. This makes the correspondence between the various 802 standards and their applicable conformance test requirements readily apparent. Thus the conformance test documents for 802.3 are numbered 1802.3, the conformance test documents for 802.5 will be 1802.5, and so on. Similarly, ISO will use 18802 to number conformance test standards for 8802 standards.

IEEE Std 802.1E, 1994 Edition

This standard defines services and protocol elements that permit the downloading of memory images to data processing equipment attached to IEEE 802 Local and Metropolitan Area Networks. The protocol makes use of the group addressing capabilities inherent in LAN/MAN technologies to permit simultaneous loading of the same memory image to multiple destination systems. The standard includes the specification of managed objects that permit the operation of the load protocol to be remotely managed; these specifications, along with the Protocol Implementation Conformance Statement (PICS) proforma in annex A, were developed as P802.1m/D4, a supplement to IEEE Std 802.1E.

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material.

Information on the current revision state of this and other IEEE 802 standards may be obtained from:

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

IEEE 802 committee working documents are available from:

IEEE Document Distribution Service
AlphaGraphics #35 Attn: P. Thrush
10201 N. 35th Avenue
Phoenix, AZ 85051, USA

Participants

The following is a list of participants in the IEEE Project 802.1 Working Group at the time IEEE Std 802.1E-1990 was approved:

William P. Lidinsky, *Chair**
Tony Jeffree, *Chair, Network Management Task Group**

Fumio Akashi
Paul D. Amer
Charles Arnold
Floyd Backes*
Ann Ballard
Richard Bantel
Sy Bederman
Amatzia Ben-Artzi
Robert Bledsoe
Kwame Boakye
Juan Bulnes
Fred Burg
Peter Carbone
Alan Chambers*
Ken Chapman
Alice Chen
Michael Chernick
Jade Chien
Jim Corrigan
Paul Cowell*
Peter Dawe
Stan Degen*
Frank Deignan
Desh Deshpande
Ron Dhondy
Kurt Dobbins
Eiji Doi
Barbara J. Don Carlos
Walter Eldon
Eldon D. Feist
Len Fishler*
Kevin Flanagan*
Pat Gonia
Richard Graham*
Michael A. Gravel
Mogens Hansen
Harold Harrington
John Hart*
Mike Harvey*
Bob Herbst
Jack R. Hung
Thomas Hytry
Jay Israel
Ram Kedlaya
Hal Keen*
Alan Kirby
Kimberly Kirkpatrick
Steve Kleiman
James Kristof*
H. Eugene Latham*
Chao-yu Liang
Bing Liao*
George Lin*
Mike Lumpkin
Andy Luque
Phillip Magnuson
Bruce McClure
Tom McGowan
Margaret A. Merrick
Jim Montrose
Jerry O'Keefe
Richard Patti*
Roger Pfister*
Thomas L. Phinney
David Piscitello
Daniel Pitt*
Ron L. G. Prince
Nigel Ramsden
Trudy Reusser
Edouard Rocher
Paul Rosenblum*
John Salter
Alan Sarsby
Susan Schanning
Mick Seaman*
Gerry Segal*
Rich Seifert*
Howard Sherry
Wu-Shi Shung
M. Soha
Dan Stokesberry
Lennart Swartz
Kenta Takumi
Robin Tasker*
Angus Telfer
Dave Thompson
Nathan Tobol
Wendell Turner
Peter Videcrantz*
Paul Wainright
Scott Wasson*
Daniel Watts
Alan Weissberger
Deborah Wilbert
Igor Zhovnirovsky*
Carolyn Zimmer*
Nick Zucchero

* Voting member of the 802.1 Working Group at the time of approval of this document.

The following persons were on the balloting committee of IEEE Std 802.1E-1990:

Willian B. Adams
Hasan S. Alkhatib
Kit Athul
Willian E. Ayen
Eduardo W. Bergamini
Peter I. P. Boulton
Paul W. Campbell
George S. Carson
Chih-Tsai Chen
Michael H. Coden
Robert Crowder
Mitchell G. Duncan
John E. Emrich
Philip H. Enslow
John W. Fendrich
Donald A. Fisher
Harvey A. Freeman
Ingrid Fromm
D. G. Gan
Patrick Gonia
Michael D. Graebner
Maris Graube
J. Scott Haugdahl
Richard J. Iliff
Raj Jain
Tony Jeffree
K. H. Kellermayr
Samuel Kho
Thomas M. Kurihara
Anthony B. Lake
Mike Lawler
Jaiyong Lee
William Lidinsky
F. C. Lim
Randolph Little
William D. Livingston
Donald Loughry
Andy J. Luque
Kelly C. McDonald
Richard H. Miller
David S. Millman
John E. Montague
Kinji Mori
Charles Oestereicher
Young Oh
Rafat Pirzada
Udo W. Pooch
John P. Riganati
Gary S. Robinson
Robert Rosenthal
Norman Schneidewind

Adarshpal S. Sethi	Efstathios D. Sykas	Donald F. Weir
D. A. Sheppard	Daniel Sze	Alan J. Weissberger
Leo Sintonen	Nathan Tobol	Earl J. Whitaker
John Spragins	L. David Umbaugh	George B. Wright
Carel M. Stillebroer	Thomas A. Varetoni	Oren Yuen
Fred J. Strauss	James T. Vorhies	Lixia Zhang

When the IEEE Standards Board approved IEEE Std 802.1E-1990 on May 31, 1990, it had the following membership:

Marco W. Migliaro, *Chair* **James M. Daly,** *Vice Chair*
Andrew G. Salem, *Secretary*

Paul L. Borrill	Kenneth D. Hendrix	Lawrence V. McCall
Fletcher J. Buckley	John W. Horch	L. Bruce McClung
Allen L. Clapp	Joseph L. Koepfinger*	Donald T. Michael*
James M. Daly	Irving Kolodny	Stig Nilsson
Stephen R. Dillon	Michael L. Lawler	Roy T. Oishi
Donald C. Fleckenstein	Donald C. Loughry	Gary S. Robinson
Jay Forster*	John E. May, Jr.	Terrance R. Whittemore
Thomas L. Hannan		Donald W. Zipse <

*Member Emeritus

IEEE Std 802.1E-1990 was approved by the American National Standards Institute (ANSI) on October 12, 1990.

The following is a list of participants in the IEEE Project 802.1 Working Group at the time IEEE Std 802.1m-1993 was approved:

William P. Lidinsky, *Chair**
Tony Jeffree, *Chair, Network Management Task Group**

Floyd Backes*	Jeanne Haney*	Ronald Presti*
Robert Barrett*	John Hart*	Maurice Qureshi
Amatzia Ben-Artzi	Mike Harvey	Paul Rosenblum
Anthony Berent*	Eli Herscovitz	Paul Ruocchio*
Orna Berry*	Bonnie Hromis*	Tom Rutt*
Sai Boeker	Long Huang	Mick Seaman*
Brian Bortz	Jan-Olof Jemnemo*	Gerry Segal
Laura Bridge*	Albert Juandy	Rich Seifert
Brian Brown	George Kajos	Steve Senum
Alan Chambers*	Hal Keen*	Himanshu Shah*
Steve Cooper*	Yoav Kluger	W. Earl Smith
Paul Cowell*	Hans Lackner	Robin Tasker*
Andy Davis*	Choon Lee	Surya Varanasi
Mike Dickerson*	George Lin	Peter Videcrantz
Len Fishler*	Joseph E. Massery	Donald G. Vincent
Daryl Francis	Alan Oppenheimer*	Trevor Warwick
Bill Futral	Richard Patti*	Richard Watson*
Lionel Geretz*	David T. Perkins*	Bernd Widmann
Richard Gilbert	Brian Phillips*	Bert Williams
Harry Gold*	John Pickens*	Jerry A. Wyatt*
Andrew Green	Venkat Prasad	Amnon Yacoby*
Sharam Hakimi*		Carolyn Zimmer

* Voting member of the 802.1 Working Group at the time of approval of this document.

The following persons were on the balloting committee of IEEE Std 802.1m-1993:

William B. Adams
Ian F. Akyildiz
Kit Athul
William E. Ayen
Paul W. Campbell
Anthony L. Carrato
John Cronican
Robert Crowder
Jose A. Cueto
Paul Eastman
John E. Emrich
Philip H. Enslow
John W. Fendrich
Harold C. Folts
Robert Gagliano
Patrick Gonia
Julio Gonzalez-Sanz
Russell D. Housley
William J. Hunteman
Richard J. Iliff
Thomas J. Jasinski
Tony Jeffree
Youngbum Kim
Dae Young Kim
Kenneth Kung
Lak Ming Lam
Michael Lawler
Lanse M. Leach
Jai Yong Lee
Randolph S. Little
Joseph Loo
Donald C. Loughry
Wen-Pai Lu
Gottfried W. R. Luderer
William C. McDonald
Ann Miller
Richard H. Miller
David S. Millman
C. B. Madhab Mishra
W. Melody Moh
John E. Montague
Kinji Mori
Alok C. Nigam
Ellis S. Nolley
Charles Oestereider
Young Oh
Thomas L. Phinney
John Pickens
Art J. Pina
Rafat Pirzada
Vikram Punj
Yang Qianli
Thad L. D. Regulinski
Francisco J. Restivo
Alan B. Richstein
John P. Riganati
Philip T. Robinson
Daniel Rosich
Floyd E. Ross
Victor Rozentouler
Christoph Ruland
Tom Rutt
Dave Sanford
Donald A. Sheppard
Dan Shia
Surindra Singh
William R. Smith
Carel M. Stillebroer
Fred J. Strauss
Efstathiois D. Sykas
Hao Tang
Patricia Thaler
Geoffrey O. Thompson
Robert Tripi
Mark-Rene Uchida
James T. Vorhies
Barry M. Vornbrock
Donald F. Weir
Raymond Wenig
Earl J. Whitaker
Jerry A. Wyatt
Oren Yuen

The final conditions for approval of IEEE Std 802.1m-1993 were met on September 29, 1993. This standard was conditionally approved by the IEEE Standards Board on June 17, 1993, with the following membership:

Wallace S. Read, *Chair* **Donald C. Loughry,** *Vice Chair*
Andrew G. Salem, *Secretary*

Gilles A. Baril
José A. Berrios de la Paz
Clyde R. Camp
Donald C. Fleckenstein
Jay Forster*
David F. Franklin
Ramiro Garcia
Donald N. Heirman
Jim Isaak
Ben C. Johnson
Walter J. Karplus
Lorraine C. Kevra
E. G. “Al” Kiener
Ivor N. Knight
Joseph L. Koepfinger*
D. N. “Jim” Logothetis
Don T. Michael*
Marco W. Migliaro
L. John Rankine
Arthur K. Reilly
Ronald H. Reimer
Gary S. Robinson
Leonard L. Tripp
Donald W. Zipse

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall
Richard B. Engelman
David E. Soffrin
Stanley I. Warshaw

Kristin M. Dittmann
IEEE Standards Project Editor

IEEE Std 802.1m-1993 was approved by ANSI on February 24, 1994.

Contents

Information technology—
Telecommunications and information exchange between systems—
Local and metropolitan area networks—
Common specifications—

Part 4: System load protocol

1. Overview

Stations in a network may, from time to time, require a portion of their local addressable memory space to be loaded using information held on a remote station. In multivendor networks, the provision of standardized mechanisms to achieve this functionality is desirable.

In order to accommodate efficient loading of multiple stations with the same information at the same time, it is desirable to provide facilities for performing the load process on a multicast basis as well as on a point-to-point basis. The System Load Protocol provides both capabilities.

The protocol assumes two types of device in any load operation:

a) Loadable Device (LD), which is capable of accepting a load from a Load Server.
b) Load Server (LS), which is capable of providing a load for a Loadable Device.

A load operation can be initiated by

a) An LD requesting data from an LS.
b) A third party requesting an LD to accept data by means of a management request, using the Load operation specified in clause 9, and specifically in 9.2.1.3, in the description of the Load operation. When an LD accepts such a request, it will then request data from an LS in the normal manner.

The loadable data is termed an image. An image is broken down into Groups that, in turn, consist of blocks. The protocol allows flexibility in the choice of image and block size. It does not state the number of Groups in an image or the number of octets in a block.

The System Load Protocol relies on IEEE 802.2 Logical Link Control (LLC) Type 1 services (see ISO 8802-2[1]) operating over any compatible MAC and Physical Layer.

The System Load Protocol allows the use of IEEE 802.1B LAN/MAN Management [ISO/IEC DIS 15802-2] to manage aspects of its operation. This use is described in 9.3. In addition, the managed objects have been defined in such a way that it is also possible to use CMIP (ISO/IEC 9596) as the management protocol, as described in 9.4.

[1]Information on references can be found in clause 3.

The System Load Protocol may be used in conjunction with other management protocols. It provides a load capability not provided by general-purpose management protocols. The general-purpose management protocols provide the capabilities of parameter manipulation, event reporting, and action invocation that support and enhance the load facility. For example, a load of one system might be invoked from another system by means of management intervention.

The following clauses describe

a) The System Load architecture.
b) The services provided by System Load.
c) The semantics and syntax of the System Load Protocol, including the state machines that describe the operation of the System Load Protocol machine.
d) The semantics of the load-related management objects.

Annex C provides further information regarding the application of the protocol.

To evaluate conformance of a particular implementation, it is necessary to have a statement of which capabilities and options have been implemented for a given protocol. Such a statement is called a Protocol Implementation Conformance Statement (PICS). Annex A to this International Standard contains the PICS proforma for the System Load Protocol.

2. Scope

The System Load Protocol specification defines a protocol to load the memory of data processing equipment installed on IEEE 802 networks. To this end the specification

a) Defines the protocol data unit (PDU) formats for loading an end system.
b) Defines the protocol for loading an end system.
c) Describes the services required of the end system being loaded (Loadable Device or LD) to successfully complete a load operation.
d) Describes the services required of the end system providing the load (Load Server or LS) to successfully complete a load operation.
e) Defines the semantic aspects of managed objects of the LD and LS that permit the manipulation of the operational parameters of the LD and LS state machines, announcement of load servers, and the third-party initiation of a load.
f) Defines the syntax used when performing management operations via the IEEE 802.1B LAN/MAN Management Protocol (ISO/IEC DIS 15802-2).
g) Defines the syntax used when performing management operations via the CMIP Systems Management Protocol (ISO/IEC 9596-1).

The specification discusses the LS in only as much detail as is necessary to define the loading protocol. Management and LS decisions, such as what to do as a result of LS or LD events or when an LS or Manager fails, are LS and Manager implementation issues and are beyond the scope of this document.

The protocol is specified to convey images consisting of data (in blocks) of unspecified format. The content and format of data blocks are application-specific; this standard does not place any constraints on

a) The form, content, or meaning of the images that may be conveyed by means of the protocol;
b) The manner in which data blocks are processed subsequent to their being received by a loadable device.

This International Standard provides the PICS proforma for the System Load Protocol in compliance with the relevant requirements, and in accordance with the relevant guidance, given in ISO/IEC 9646-2.

3. References

The following standards contain provisions which, through references in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

IEEE Std 802-1990, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture (ANSI).[2]

IEEE Std 802.1F-1993, IEEE Standards for Local and Metropolitan Area Networks: Common Definitions and Procedures for IEEE 802 Management Information.

ISO 7498-4 : 1989, Information technology—Open Systems Interconnection—Basic Reference Model—Part 4: Management framework.[3]

ISO 8802-2 : 1989 [ANSI/IEEE Std 802.2-1989], Information processing systems—Local area networks—Part 2: Logical link control.

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

ISO/IEC 8825 : 1990, Information technology—Open Systems Interconnection—Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).

ISO/IEC 9595 : 1991, Information technology—Open Systems Interconnection—Common management information service definition.

ISO/IEC 9596-1 : 1991, Information technology—Open Systems Interconnection—Common management information protocol—Part 1: Specification.

ISO/IEC 9646-1 : 1991, Information technology—Open Systems Interconnection—Conformance testing methodology and framework—Part 1: General concepts.

ISO/IEC 9646-2 : 1991, Information technology—Open Systems Interconnection—Conformance testing methodology and framework—Part 2: Abstract test suite specification.

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

ISO/IEC TR 10178, Information technology—Telecommunications and information exchange between systems—The structure and coding of Logical Link Control addresses in Local Area Networks.

ISO/IEC TR 10735, Information technology—Telecommunications and information exchange between systems—Standard Group MAC Addresses.

---

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

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

ISO/IEC DIS 15802-2…[4] [ANSI/IEEE Std 802.1B-1992 and IEEE Std 802.1k-1993], Draft Standard for Local and Metropolitan Area Networks: LAN/MAN Management.

4. Definitions

The following terms are used throughout this International Standard in a specialized manner:

4.1 Image: The data structure contained in the Load Server that the Loadable Device wishes to load.

4.2 Loadable Device (LD): A station on the network that is capable of accepting a load from a Load Server.

4.3 Load Server (LS): A station on the network that is capable of providing a load for a Loadable Device.

This International Standard uses the following terms defined in ISO/IEC 9646-1:

a) PICS proforma
b) protocol implementation conformance statement (PICS)
c) static conformance review

Other management-specific terms are defined in IEEE 802.1B LAN/MAN Management (ISO/IEC DIS 15802-2).

5. Authority

In cases where there are discrepancies between the text descriptions, state diagrams, and the state tables, the state tables are to be considered as the authority.

6. Architecture

This clause contains an overview of the System Load architecture. Figure 1 depicts the architectural components and interfaces. There are three major architectural components involved in System Load:

a) System Load User (SLU)
b) System Load Entity (SLE)
c) System Load Layer Management Entity (SL_LME)

A brief description of each of these components follows:

The SLU is a user of System Load Services. The user of these services may be a provider of load images (LS) or a requester of load images (LD) or both.

In order to perform its function, the SLU may be required to communicate with an SLU in another station. The SLUs intercommunicate in a peer-to-peer manner, using the communication services provided by the SLE. This communication is described in terms of the roles that each SLU and associated SLE plays in a particular instance of communication, either the role of an LD or an LS. These terms are used for descriptive purposes only and do not imply system capabilities in any other context. An LD requests to be loaded with an image. An LS acts on behalf of the requesting LD to provide an image.

[4]Currently at state of Draft International Standard.

The System Load Service is provided by means of the System Load Protocol. Two architectural components are associated with the operation of the protocol. These are the SLE and the SL_MIB.

The SLE performs System Load communication functions on behalf of its SLU.

The SL_MIB is a set of managed objects associated with the SLE. It provides the management functions and management information specific to the management of the SLE, allowing it to be managed in a manner analogous to managing a layer. The managed object class definitions for the SL_MIB are contained in clause 9. IEEE 802.1B [ISO/IEC DIS 15802-2] describes how the functionality of these managed objects may be accessed by means of management protocol.

The SLE makes use of the underlying services of IEEE 802 LLC Type 1 (ISO 8802-2) in order to convey SL_PDUs. The use of other services is not precluded; however, this standard does not specify any conformance-related aspects of the use of any other services.

The load-specific services of the SLE are described in clause 7.

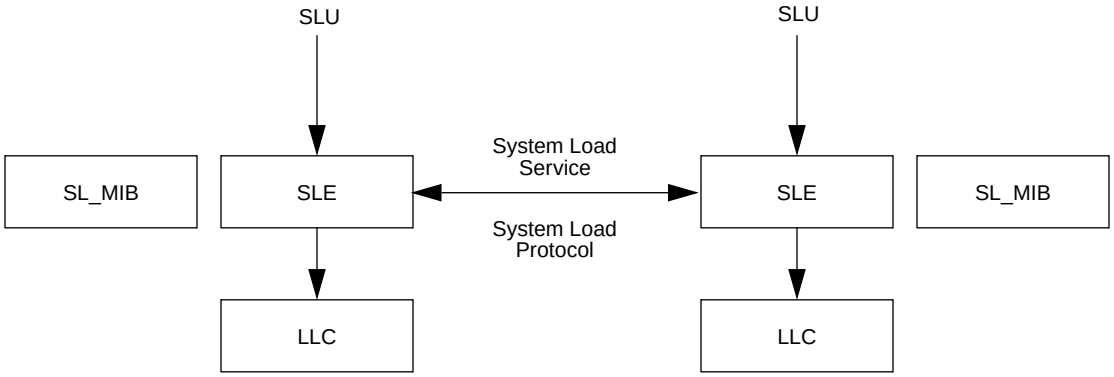


Figure 1—System Load architecture

7. Service definition

This clause defines the services provided by the System Load Entity (SLE) to the System Load User (SLU) at the System Load Service Boundary.

The following primitives are defined for the SLU to request service from the SLE:

a) SYSTEM_LOAD.request, by which the SLU requests that a load operation be performed.
b) SYSTEM_LOAD.confirm, by which the SLE confirms the success or failure of the corresponding request.
c) SYSTEM_LOAD.indication, by which the SLE informs the SLU that an image is required in order to perform a requested load operation.
d) SYSTEM_LOAD.response, by which the SLU returns the required image or reason for not providing the image.

The time sequence of these service primitives is shown in figure 2.

In addition, general-purpose management services provided by IEEE 802.1B LAN/MAN Management (ISO/IEC DIS 15802-2) and the CMIS/CMIP standards (ISO/IEC 9595 and 9596) support the load facility:

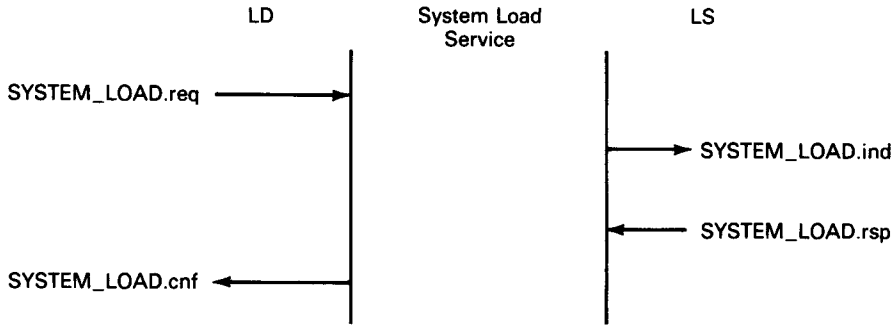


Figure 2—Sequence of SYSTEM_LOAD primitives

— Parameters affecting the operation of the protocol may be manipulated.
— Notification of events such as the availability of an LS may be effected.
— Actions such as the initiation of the load of a system may be effected.

NOTE—The definition of the System Load Service in terms of these primitives and their associated parameters is for clarity only, and should not be construed as placing constraints on the nature of real implementations.

7.1 SYSTEM_LOAD.request

This primitive is the service request primitive for the System Load Service.

7.1.1 Semantics of the service primitive

```
SYSTEM_LOAD.request   (
                      load_info,
                      load_reason
                      )
```

— load_info specifies information that may be used to determine the image to be loaded.
— load_reason specifies the reason that the load is desired. The load reason parameter shall have one of the following values:

Unspecified:	No reason is specified.
PowerUp:	The system or some part of it has had power restored.
ForcedLoad:	The load has been forced remotely via a systems management action.
OperationalFailure:	An operational failure has occurred resulting in the necessity for a load.
LoadFailure:	A previous load attempt has failed.
Reconfiguration:	A configuration change requiring load has occurred.
Private Reason:	An implementation-specific reason.

7.1.2 When generated

This primitive is used by the SLU whenever it needs to have part or all of its system loaded.

7.1.3 Effect of receipt

The effect of this primitive is that the SLE shall request the load from one or more load servers.

7.2 SYSTEM_LOAD.indication

This primitive is the service indication primitive for the System Load Service.

7.2.1 Semantics of the service primitive

```
SYSTEM_LOAD.indication  (
                        load_info,
                        load_reason
                        )
```

— load_info specifies information that may be used to determine the image to be loaded. The load_info parameter shall take the value provided in the corresponding SYSTEM_LOAD.request primitive.
— load_reason specifies the reason that the load is desired. The load_reason parameter shall take the value provided in the corresponding SYSTEM_LOAD.request primitive.

7.2.2 When generated

This primitive is generated upon receipt of a LoadRequestPDU by the SLE.

7.2.3 Effect of receipt

The effect of this primitive is that the SLU shall return the required image or reason for not returning the image in a SYSTEM_LOAD.response primitive.

7.3 SYSTEM_LOAD.response

This primitive is the service response primitive for the System Load Service.

7.3.1 Semantics of the service primitive

```
SYSTEM_LOAD.response    (
                        status,
                        image
                        )
```

— status specifies the success or failure of the load request. The status parameter shall take one of the following values:

success:	The image is available and supplied.
not_available:	The image is not available for the indicated load.

— image is the image to be provided by the load; it is not provided if status is not success.

7.3.2 When generated

This primitive is used by the SLU to return the image indicated in the corresponding SYSTEM_LOAD.indication primitive.

7.3.3 Effect of receipt

The effect of this primitive is that the SLE shall attempt to perform the load.

7.4 SYSTEM_LOAD.confirm

This primitive is the service confirmation primitive for the System Load Service.

7.4.1 Semantics of the service primitive

```
SYSTEM_LOAD.confirm  (
                       status,
                       image
                     )
```

— status specifies the success or failure of the load request. The status parameter shall take one of the following values:

success:	The load was correctly performed.
no_response:	No server responded to the request.
incomplete:	A load was attempted but never successfully completed.
invalid_response:	An invalid load response(s) was (were) received with parameters inconsistent with the corresponding load request.

— image is the image provided by the load; it is not provided if status is not success.

7.4.2 When generated

This primitive shall be generated by the SLE upon completion of performing or attempting to perform the load requested; it returns the success or failure of the request and the image requested (if successful).

7.4.3 Effect of receipt

Unspecified.

8. Protocol specification

8.1 Summary of Protocol Data Units (PDUs)

The protocol is described in terms of Load PDUs exchanged between an LD and an LS.

— A LoadRequestPDU is sent from an LD to an LS or an LS group address to request a load image.
— A LoadResponsePDU is sent from an LS to an LD or an LD group address to inform the LD(s) that the LS is willing to send a load image.
— A GroupStatusPDU is sent from an LD to an LS in order to request portions of a load image.
— A GroupStatusRequestPDU is sent from an LS to an LD or an LD group address in order to ask an LD or LDs if portions of a load image are needed.
— A LoadDataPDU is sent from the LS to an LD or an LD group address. It contains a portion of a load image.

The process of loading an image is performed by transmitting the image as a number of Blocks, each of which is of a fixed size for a particular load operation, except, possibly, for the last Block of the load. Each set of 256 consecutive Blocks is defined to be a Group. The last Group of the image may contain less than 256 Blocks, and the last Block of the image may be only partially full. The number of Groups and Blocks in a given image is therefore calculated as follows:

$$\text{Number of Blocks} = \frac{\text{Image size} + \text{Block size} - 1}{\text{Block size}}$$

$$\text{Number of Groups} = \frac{\text{Number of Blocks} + 255}{256}$$

NOTE—The equations for Number of Blocks and Number of Groups assume integer arithmetic; integer division will discard any remainder.

The following subclauses describe the content and semantics associated with each PDU.

8.2 LoadRequestPDU

8.2.1 Function

The function of the LoadRequestPDU is to request a system load from an LS or a set of LS's. It is issued by the LD.

8.2.2 Semantics

The following describes the fields of the LoadRequestPDU and the associated semantics:

ExchangeID

This field is an identifier that the LD may use to uniquely identify a load request. The value is determined by the LD. This field is optional and need only be used if the LD has multiple simultaneous outstanding requests.

LoadAddress

This field specifies the MAC address to which the LD requires the LS to send the load data. If present, the address shall be either the station's own individual MAC address or a group MAC address. If not present, the LS shall choose either the individual MAC address of the LD, or a group MAC address.

NOTE—The LD has the following options with respect to the choice of address on which it is prepared to accept PDUs from the LS. They are as follows:

a) If LoadAddress specifies the specific MAC address, then the LD is only prepared to accept PDUs on this address.
b) If LoadAddress specifies a group MAC address, the LD will not recognize any other group MAC address; however, the LS may choose to use the LD's individual MAC address or the indicated group MAC address at its own discretion.
c) If LoadAddress is not present, the LD is prepared to accept PDUs addressed to its individual MAC address or any group MAC address of the LS's choosing. If the LS chooses to use the individual MAC address of the LD, this is derived from the addressing information provided by the underlying service.

BlockSize

This field specifies the maximum data block size in octets which the station can accept for loading. It is the number of octets in the LoadData field of the LoadDataPDU. It ranges from 1 to 32 767.

MinBlockDelay

This field specifies the minimum delay in milliseconds required between transmission of data blocks in order to minimize errors due to depletion of resources at the LD. It ranges from 0 to 32 767. The consequences, if data blocks are received with less than MinBlockDelay separation between them, are unspecified.

MaxBlockDelay

This optional field specifies the maximum amount of delay in milliseconds which the LD will tolerate between transmission of individual data blocks. It shall be larger than MinBlockDelay but less than or equal to 32 767. If not provided, the maximum value shall be assumed. This value should generally be specified as large as possible, providing the LS with the maximum flexibility in “merging” load requests. The consequences, if MaxBlockDelay is exceeded, are unspecified.

LoadReason

This optional field contains the reason for sending the LoadRequestPDU. The value of this field shall be the same as that supplied in the load_reason parameter of the associated SYSTEM_LOAD.request primitive. If not provided, a value of Unspecified shall be assumed.

LoadInfo

LoadInfo is an optional field of the LoadRequestPDU. It may be used to describe the equipment that is requesting the load and the image requested.

LoadInfo is subdivided into PrivateID, StationID, and ImageID. All three fields are optional.

PrivateID is unadministered and is available for implementation-specific use.

StationID is a substructured field that contains information describing the device that is requesting the load. It contains an administered field, ManufacturerID, and two unadministered fields, DeviceTypeID and RevisionNumber. Each of the three fields are optional. The ManufacturerID is an Organizationally Unique Identifier (OUI) administered by the IEEE, as described in 5.2 of IEEE Std 802-1990. When present, the ManufacturerID provides the context within which the unadministered fields of LoadInfo may be interpreted.

ImageID is an unadministered field that contains information describing a requested load image. ImageID may be an explicit image name, a symbolic reference to one or more images, or some other implementation-specific form.

The LS may use the information provided in LoadInfo in order to determine the image to send to the requesting device. The LS may return some or all LoadInfo fields to the requesting LD as well as additional implementation-specific information.

8.2.2.1 When generated

The LoadRequestPDU is generated by an LD upon determining that it needs to obtain a load. An LD may make this determination itself (by unspecified means) or be instructed to do so by a general network manager (via the action service of the general-purpose systems facility).

8.2.2.2 Effect of receipt

On receipt of a LoadRequestPDU, an LS determines whether it can provide the load of the requesting LD. If so, the LS sends a LoadResponsePDU, then waits for the LD to send a GroupStatusPDU. If the LS cannot load the requesting LD, it does not send a LoadResponsePDU.

8.3 LoadResponsePDU

8.3.1 Function

The function of the LoadResponsePDU is to respond positively to one or more LoadRequestPDUs.

8.3.2 Semantics

The following describes the fields of the LoadResponsePDU and the associated semantics:

ExchangeID

This field is used by the LD to identify the corresponding request. If the LoadRequestPDU contained an ExchangeID, the LoadResponse PDU shall contain an ExchangeID with the same value as that contained in the LoadRequestPDU.

LoadAddress

This field specifies the MAC address to which the LS will send subsequent LoadPDUs of the requested load. If this field is absent from the LoadRequestPDU, the LS shall specify the MAC address to be used. See also 8.2.2, LoadAddress.

BlockSize

This specifies the size of the data blocks that the LS will send. It is the number of octets in the LoadData field of the LoadDataPDU. It ranges from 1 to 32 767.

MinBlockDelay

This field specifies the minimum amount of time in milliseconds which the LD can expect between transmissions of successive data blocks.

MaxBlockDelay

This field specifies the maximum amount of delay in milliseconds that the LD can expect between transmission of individual data blocks. It shall be larger than MinBlockDelay. It shall include not only the maximum delay that the LS may insert, but also the maximum delay expected to be introduced by the network.

The minimum possible value of MaxBlockDelay will result in the most rapid detection of errors. The value should be the sum of the MinBlockDelay, the expected “poor case” LS implementation delay, and the expected “poor case” network delay, which may be available in the LSNetDelay parameter (see LS parameter definitions).

NOTE—Successive data block transmissions should be separated by [MinBlockDelay + LSNetDelay], since if one PDU is delayed by the maximum and the next delayed not at all, they may arrive closer together than MinBlockDelay. Clearly, the returned value of MaxBlockDelay should be greater than or equal to the returned value of MinBlockDelay.

ReferenceID

This field specifies an identifier provided by the LS that is used to identify all subsequent PDUs associated with this load operation. This may be any 16-bit value.

NOTE—This identifier uniquely identifies a particular load in the context of a given LS.

NumberBlocks

This field specifies the total number of data blocks associated with this load operation. It ranges from 1 to 65 535.

LoadSelector

The LoadSelector is an optional field that may be used by the LD to choose between multiple eligible LS load "offers." The LS may assign this field any value in the range of –128 to 127. See 8.3.3.

ImageInfo

ImageInfo is an optional field that describes the load image the LS will send. It may contain any or all of the LoadInfo fields sent in the LoadRequestPDU as well as additional information.

8.3.2.1 When generated

The LoadResponsePDU is generated by the LS after determining that it is prepared to provide the load of an LD from which it has received a LoadRequestPDU. If the LS is not already performing a load of the requested image, after sending the LoadResponsePDU, the LS waits for a GroupStatusPDU before proceeding with the load.

8.3.2.2 Effect of receipt

The LD may receive more than one LoadResponsePDU if there is more than one LS present that is prepared to load it. The LD selects an LS from those that send a LoadResponsePDU, by sending a GroupStatusPDU to the LS that is to provide the load. The LD then waits for subsequent load-related PDUs from the selected LS.

The LD uses the value of NumberBlocks to calculate the number of Groups in the load (see 8.1). The image is sent from the LS in groups. There may be up to 256 Groups (Group number 0 to 255). Except for the last Group, there are 256 blocks within each Group (block number 0 to 255). The 256-block Group size is consistent with the size of the bitmap described in 8.4.2.

8.3.3 Additional comments

When more than one LS offers to provide the load with equally acceptable parameters (e.g., MaxBlockDelay, ImageInfo), the LD may choose the LS that sent the highest LoadSelector value. The LD may make an arbitrary choice of eligible LS's in the event of a tie. The mechanism whereby the LS decides on a value of LoadSelector is not defined by this standard.

This field may be used in order to facilitate load sharing as well as to discourage multiple simultaneous instances of the same load. Refer to annex C.

8.4 GroupStatusPDU

8.4.1 Function

A GroupStatusPDU is sent by the LD to the LS in order to request the load of specific blocks. The LD sends a GroupStatusPDU to the individual MAC address of the LS that is to provide the load. Thereafter, the LD sends GroupStatusPDUs in order to request the retransmission of blocks.

8.4.2 Semantics

The following describes the fields of the GroupStatusPDU and the associated semantics:

ReferenceID

This field is the identifier provided by the LS in the LoadResponsePDU.

GroupNumber

This optional field indicates to which Group number the requested blocks belong. It ranges from 0 to 255. If this field is absent, a default value of “all groups” is assumed by the LS.

RequiredBlocks

This field specifies which blocks from the specified Group number are required by this LD. There are two possible forms:

— RequiredBlocks may be a 256-bit map (BitMap) in which each “set” bit corresponds to a required block from the specified GroupNumber. This form may occur only if GroupNumber is present. The BitMap is encoded with the most significant bit as bit 0, corresponding to block number 0 in a field of 8 bits.
— It may indicate that all blocks or no blocks of a load image or Group are needed for reassembly (RequiredBlocksCode).

8.4.2.1 When generated

The LD sends a GroupStatusPDU to the individual MAC address of the LS that is to provide the load.

If the LD needs all the blocks of the load image, it sends a GroupStatusPDU with no GroupNumber field but with a RequiredBlocks field that indicates that all blocks are needed.

If the LD needs only a partial load, it sends a GroupStatusPDU for each Group of the load image for which it requires blocks. Thereafter, the LD sends GroupStatusPDUs to the LS in order to request the retransmission of blocks.

A GroupStatusPDU shall be generated at the end of each Group during the load if there are required blocks outstanding.

At least one GroupStatusPDU shall be sent in response to a GroupStatusRequestPDU if there are required blocks outstanding.

The LD may send multiple GroupStatusPDUs, if required, without delay between transmissions.

8.4.2.2 Effect of receipt

If the LS is not already performing a load of the requested image, after sending a LoadResponsePDU, the LS waits for a GroupStatusPDU. If no GroupStatusPDU is received, the LS does not proceed with the load. If a GroupStatusPDU is received, the LS records the blocks needed for the load and proceeds with their transmission.

If the LS receives a GroupStatusPDU for the reference ID of the load in progress, it merges the blocks of the load image indicated in the GroupStatusPDU with the blocks it has left to send.

8.5 GroupStatusRequestPDU

8.5.1 Function

A GroupStatusRequestPDU is issued by the LS to determine if the LDs involved in the load require blocks of the load image. The GroupStatusRequestPDU may be sent to the individual MAC address of an LD or sent to a number of LDs by use of a group MAC address.

8.5.2 Semantics

The following describes the fields of the GroupStatusRequestPDU and the associated semantics:

ReferenceID

This field is the identifier provided by the LS in the LoadResponsePDU.

When generated

The LS shall generate the GroupStatusRequestPDU at the end of each pass through the image.

8.5.2.1 Effect of receipt

If, on receipt of a GroupStatusRequestPDU, an LD requires blocks from any Groups of the load, it returns a GroupStatusPDU for each Group for which it requires blocks. If the LD does not require blocks, it shall not reply to the GroupStatusRequestPDU.

8.6 LoadDataPDU

8.6.1 Function

The function of the LoadDataPDU is to transmit a data block from the LS to the LD(s) involved in the load.

8.6.2 Semantics

The following describes the fields of the LoadDataPDU and the associated semantics:

ReferenceID

This field is the identifier provided by the LS in the LoadResponsePDU.

GroupNumber

This field specifies to which Group number the data block belongs. It ranges in value from 0 to 255.

Block

This field contains the number of the data block within the Group which is being sent. It ranges in value from 0 to 255.

LoadData

This field contains the data to be loaded. Note that each LoadDataPDU should generally be an independent unit of information since the protocol does not guarantee delivery of LoadData PDUs in order.

8.6.2.1 When generated

The LoadDataPDU is generated by the LS at intervals such that the interval is equal to or greater than the value of MinBlockDelay in the LoadResponsePDU. LoadDataPDUs are generated until all blocks that were requested have been transmitted.

NOTE—The interval between transmissions of LoadDataPDUs from the LS shall not be less than the value of MinBlockDelay sent in the LoadResponsePDU. While this would result in an average interval between LoadDataPDUs at the LD of not less than MinBlockDelay, there will be no guarantee that LoadDataPDUs will not arrive more closely spaced as a consequence of bunching effects in the network. Where possible, the LD should be designed with sufficient buffering capability to allow for bursts of closely spaced LoadDataPDUs.

8.6.2.2 Effect of receipt

On receipt of a LoadDataPDU, an LD loads the data contained in the LoadDataPDU. The meaning of this or the mechanism whereby it is achieved are outside the scope of this protocol.

8.7 Elements of operation

This subclause provides a list of state variables, a verbal description of the loading process, state diagrams, and state tables for both the LD and the LS.

NOTE—For the purposes of this description, timers and counters are assumed to be initialized by setting them to a positive value. Thereafter, the timer or counter counts down towards zero upon occurrence of appropriate events and is said to have expired upon reaching zero, at which point the countdown terminates. This description of timer and counter operation in no way constrains real implementations of the timers or counters so described.

8.7.1 Operation of the Loadable Device (LD)

The load operation is initiated by the System Load User in an LD when it determines that a load is required, and is initiated by the SLU issuing a SYSTEM_LOAD.request primitive with appropriate parameter values. This is an internal decision made by the system concerned. It may be the result of a local condition, or a consequence of some external command or request issued by means of Management protocol. The specification of mechanisms for determining the need for a load are outside the scope of this International Standard.

The load may include the entire image or any portion thereof. The SYSTEM_LOAD.request may actually be initiated on behalf of a device connected to the station which itself needs to be loaded. The content of the LoadData field of the LoadDataPDU and the action taken by the station upon its receipt are also beyond the scope of this protocol.

The following two subclauses, 8.7.1.1 and 8.7.1.2, describe the state variables and procedures relating to the operation of the LD. The state transitions are summarized in figure 3 and table 1.

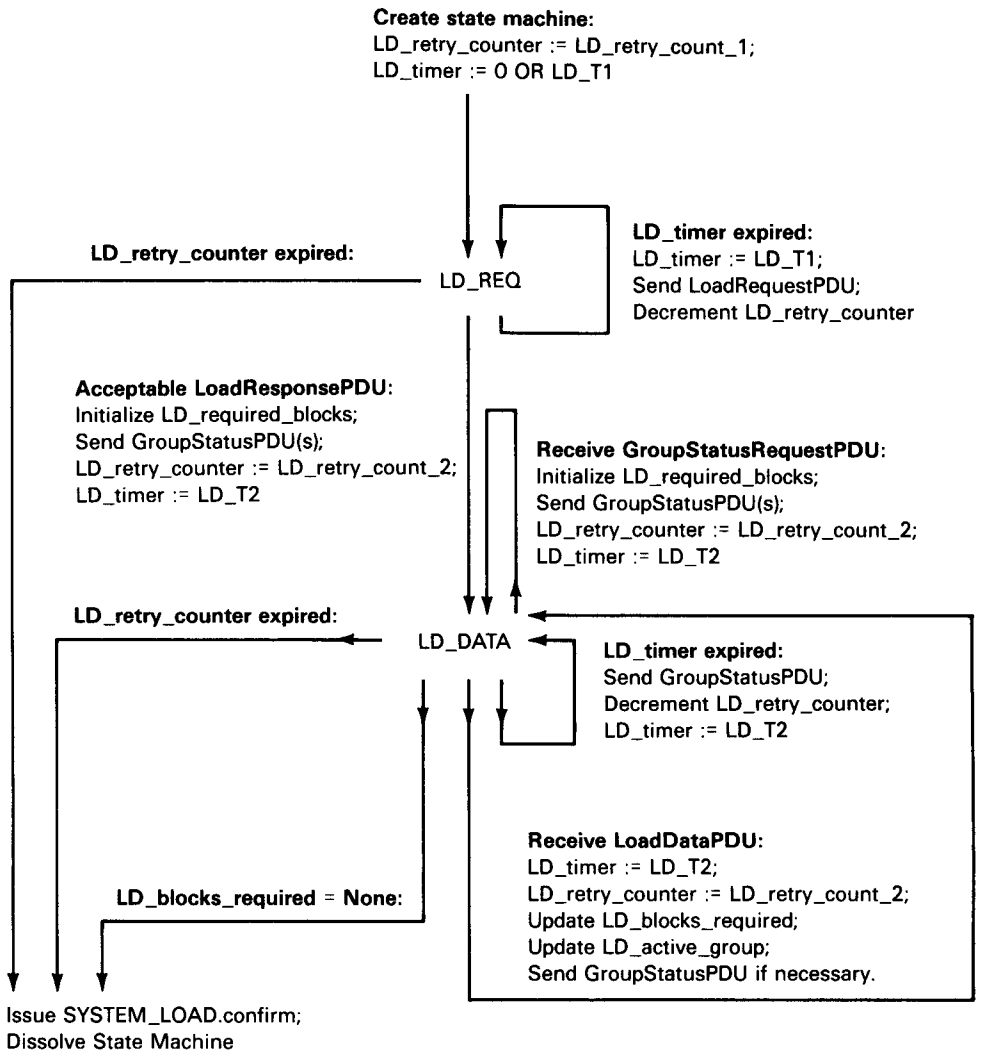
8.7.1.1 LD state variables

The LD creates a state machine in order to receive a load from an LS. The following state variables are used to describe the operation of an LD state machine.

LD_retry_counter

Counter used to detect conditions such as network or LS failure. It is set to the number of times an LD state machine will wait for a load-related PDU from an LS. It is initialized to one of the following:

— LD_retry_count_1 Number of LoadRequestPDUs that may be sent before the LD state machine enters the LD_failed state.



NOTE—State transitions that may be caused by the intervention of external management mechanisms are not shown on this diagram.

Figure 3—LD state diagram

— LD_retry-count_2 Number of consecutive timeout periods that the LD will wait for receipt of load-related PDUs before the LD state machine enters the LD_failed state.

LD_timer

Timer used to generate retries when expected responses are not received. It is initialized using one of the following:

— LD_T1 Length of time the LD state machine will wait for an acceptable LoadResponsePDU from an LS before reissuing the LoadRequestPDU. It allows time for the LS to accumulate requests from multiple LDs before issuing a LoadResponsePDU.

— LD_T2 Length of time the LD will wait for a LoadDataPDU (or a GroupStatusRequestPDU) before issuing a GroupStatusPDU. LD_T2 is greater than or equal to the value of MaxBlockDelay sent in the LoadResponsePDU.

LD_reference_ID

Holds the value of the ReferenceID field returned in the LoadResponsePDU and the address of the LS that returned the LoadResponsePDU.

LD_number_blocks

Holds the value of the NumberBlocks field returned in the LoadResponsePDU.

LD_number_groups

Holds the number of Groups required for this load. Determined from: (LD_number_blocks + 255)/256.

LD_active_group

Holds the Group number for which data blocks are currently being received.

LD_blocks_required [LD_number_groups]

An array, each element of which contains a set of indicators that correspond to the numbers of the data blocks that need to be loaded for a given Group. There are a total of LD_number_blocks indicators in the array, and LD_number_groups elements.

LD_status_sent

Indicates whether a GroupStatusPDU has been sent to the LS for the LD_active_group.

LD_unacceptable_LoadResponsePDU_rcvd

Indicates whether an unacceptable LoadResponsePDU has been received while in the LD_REQ state.

8.7.1.2 Description of procedures

The System Load Entity creates an instance of the LD state machine, described in the following subclauses, and places the state machine in the LD_REQ state when a SYSTEM_LOAD.request primitive is issued. LD_timer is set to a value of either 0 or LD_T1 and LD_retry_counter is set to a value of LD_retry_ count_1 before the LD_REQ state is entered.

NOTE—The initial value of LD_timer (0 or LD_T1) chosen will determine whether the state machine immediately issues a LoadRequestPDU or whether it waits for an interval of LD_T1 for a LoadResponsePDU with parameters to match its requirements. The latter choice might be made in circumstances where a high probability of simultaneous load requests exists, e.g., after a power failure.

8.7.1.2.1 LD LOAD REQUEST State (LD_REQ)

The state machine sends a LoadRequestPDU upon expiry of LD_timer. If multiple requests will be simultaneously outstanding, a unique ExchangeID is included in each request.

The LoadRequestPDU is addressed to the individual MAC address of an LS or to an LS group MAC address. It contains the BlockSize, the minimum delay required to process successive data blocks without

Table 1—LD state table

CURRENT STATE: —		
EVENT	ACTION(S)	NEXT STATE
Load needed: No state machine	Initialize state variables; LD_timer := LD T1 OR 0; LD_unacceptable_ LoadResponsePDU_rcvd := 0; LD_retry_counter := LD_retry_count_1	LD_REQ
CURRENT STATE: LD_REQ		
EVENT	ACTION(S)	NEXT STATE
Receive acceptable LoadResponsePDU	Send GroupStatusPDU to 1 LS; LD_retry_counter := LD_retry_count_2; LD_timer := LD_T2; Initialize LD_required_blocks; Initialize LD_reference_ID	LD_DATA
Receive unacceptable LoadResponsePDU	LD_unacceptable_ LoadResponsePDU rcvd := 1	LD REQ
LD_timer expired & LD_retry_counter not expired	Send LoadRequestPDU; Decrement LD_retry_counter; LD_timer := LD_T1	LD REQ
LD_retry_counter expired and LD_unacceptable_Load_ ResponsePDU_rcvd=0	Issue SYSTEM_LOAD.confirm (status = no response); Dissolve State Machine	NULL
LD_retry_counter expired and LD_ unacceptable_Load_ ResponsePDU_rcvd=1	Issue SYSTEM_LOAD.confirm (status = invalid response); Dissolve State Machine	NULL
Other	Ignore	LD REQ
CURRENT STATE: LD DATA		
EVENT	ACTION(S)	NEXT STATE
Receive LoadDataPDU with (referenceID, source address) = LD_reference_id	LD_timer := LD_T2; LD_retry_counter := LD_retry_count_2; If block needed: [load data; update LD_blocks_required; If 1st LoadDataPDU: [LD_active_group := GroupNum; LD_status_sent := 0;] If GroupNum<>LD_active_group: [If LD_blocks_required in LD active group<>None & LD_status_sent = 0, [send GroupStatusPDU with GroupNumber = LD_active_group;] LD_active_group := GroupNum; LD_status_sent := 0;]] If last block of Group but blocks required in Group & LD_status_sent = 0: [send GroupStatusPDU; LD_status_sent := 1;]	LD_DATA

Table 1—LD state table *(Continued)*

CURRENT STATE: LD DATA		
EVENT	ACTION(S)	NEXT STATE
LD_timer expires & LD_retry_counter not expired	Send GroupStatusPDU; Decrement LD_retry_counter; LD_timer := LD_T2	LD_DATA
LD_retry_counter expired	Issue SYSTEM_LOAD.confirm (status = incomplete); Dissolve State Machine	NULL
Receive GroupStatusReqPDU with (ReferenceID, source address) = LD_reference_id	Send GroupStatusPDU for each group with required blocks; LD_retry_counter := LD_retry_count_2; LD_timer := LD_T2	LD_DATA
LD_blocks_required = 0	Issue SYSTEM_LOAD.confirm (status = success); Dissolve State Machine	NULL
Other	Ignore	LD_DATA

loss, MinBlockDelay, and the maximum delay preferred, MaxBlockDelay. These need not be worst-case values since the protocol allows for efficient recovery from data block loss. The LoadInfo field may also be present in the LoadRequestPDU.

After the LoadRequestPDU is issued, LD_timer is initialized to LD_T1, and the LD listens for a LoadResponsePDU sent to its individual MAC address or a group MAC address. If no response is received in the timeout period, LD_retry_counter is decremented. If LD_retry_counter expires, a SYSTEM_LOAD.confirm primitive is issued to inform the SLU of the failure and the reason for it, and the state machine is dissolved. If LD_retry_counter is not expired, another LoadRequestPDU is issued and LD_timer is initialized to LD_T1.

When a LoadResponsePDU is received, the ExchangeID (if present), ImageInfo, BlockSize, and MinBlock-Delay fields are checked. The LoadSelector field may also be examined. More than one acceptable Load-ResponsePDU may be received. The LS that is to provide the load shall be selected from the LS's that sent an acceptable LoadResponsePDU. The LS returning the highest value of LoadSelector may be accepted if all other factors (such as MaxBlockDelay, ImageInfo, etc.) are equal. Where all factors, including LoadSelector, are equal, the LD may make an arbitrary selection of LS.

Once the LS has been chosen, the value of the ReferenceID is stored in state variable LD_reference_id.

If the entire load image is required, NumberBlocks is stored in state variable LD_number_blocks, LD_number_groups is calculated, and LD_blocks_required is initialized.

If only a portion of the load image is required, LD_number_blocks, LD_number_groups, and LD_blocks_required are initialized accordingly.

The LD state machine then sends one or more GroupStatusPDUs for the blocks required to the individual MAC address of the LS. The LS that is to provide the load is thus selected.

LD_timer is initialized to LD_T2, LD_retry_counter is initialized to LD_retry_count_2 and LD_DATA state is entered.

8.7.1.2.2 LD DATA state (LD_DATA)

Any PDUs with ReferenceID not equal to LD_reference_id shall be ignored.

On receipt of the first LoadDataPDU for an instance of the state machine, LD_active_group is set using the value of GroupNumber and LD_status_sent is cleared.

When a LoadDataPDU or a GroupStatusRequestPDU whose ReferenceID field matches the LD_reference_id state variable is received, LD_timer is initialized to LD_T2 and LD_retry_counter is set to LD_retry_count_2.

If LD_timer expires, LD_retry_counter is decremented. If LD_retry_counter has expired, a SYSTEM_LOAD.confirm is issued to inform the SLU of the failure and the reason for it, and the state machine is dissolved. Otherwise, a GroupStatusPDU is sent and LD_timer is initialized to LD_T2.

If the GroupNumber contained in a LoadDataPDU matches LD_active_group and if BlockNumber corresponds to a required block of LD_blocks_required, then the data block is loaded into memory and LD_blocks_required is updated to indicate the block has been received. If the data block is not required, it is ignored.

If a received LoadDataPDU contains a required block, but the received GroupNumber does not match LD_active_group, then

- If there are blocks still required in the active Group and LD_status_sent is not set (that is, if a GroupStatusPDU has not already been sent for the group), a GroupStatusPDU is sent for the LD_active_group.
- LD_active_group is set to the value of the received GroupNumber and LD_status_sent is cleared.
- The received data block is loaded.

LD_blocks_required [LD_active_group] is scanned on receipt of each LoadDataPDU. If there are no required blocks in LD_active_group with a block number greater than the Block field in the LoadDataPDU but there are prior required blocks in LD_active_group, and if LD_status_sent flag is not set, then a GroupStatusPDU is sent and the LD_status_sent flag is set.

If a GroupStatusRequestPDU is received from the LS while the state machine is in the LD_DATA state, LD_blocks_required is scanned, one Group at a time, for any required blocks. When required blocks are detected in a given Group, a GroupStatusPDU is sent for that Group and the scan continues until all Groups have been examined.

When all elements of LD_blocks_required have been cleared, all data blocks have been received correctly and the load is complete. A SYSTEM_LOAD.confirm primitive is issued to pass the received image to the SLU, and the state machine is dissolved.

8.7.2 Operation of the Load Server (LS)

The following two subclauses, 8.7.2.1 and 8.7.2.2, describe the state variables and procedures relating to the operation of the LS. The state transitions are summarized in figure 4 and table 2.

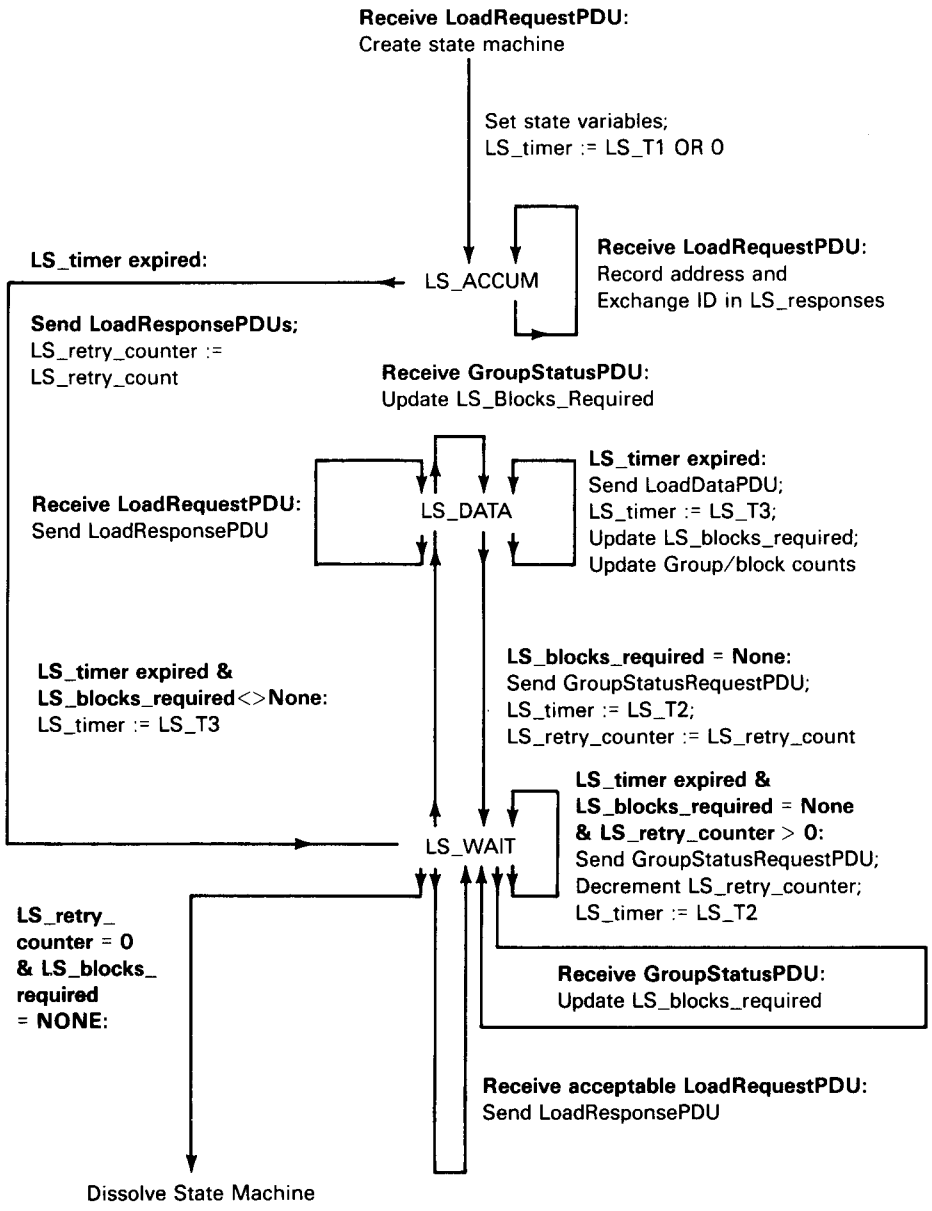


Figure 4—LS state diagram

8.7.2.1 LS state variables

The LS creates state machines in order to send load images to LDs. The following state variables are used to describe the operation of the LS state machine.

LS_load_info

Variable corresponding to the LoadInfo field from LoadRequestPDU.

LS_image_id

Identifier of the load image to be sent.

Table 2—LS state table

CURRENT STATE: —		
EVENT	ACTION(S)	NEXT STATE
Receive acceptable LoadRequestPDU; compatible state machine not active	Initialize state variables; LS_timer := LS_T1; Record address/exchange ID in LS_responses	LS_ACCUM
CURRENT STATE: LS_ACCUM		
EVENT	ACTION(S)	NEXT STATE
Receive acceptable LoadRequestPDU	Record address/exchange ID in LS_responses	LS_ACCUM
LS_timer expires	Send LoadResponsePDUs; LS_timer := LS_T2; LS_retry_counter := LS_retry_count	LS_WAIT
Other	Ignore	LS ACCUM
CURRENT STATE: LS_DATA		
EVENT	ACTION(S)	NEXT STATE
Receive acceptable LoadRequestPDU	Send LoadResponsePDU	LS_DATA
Receive GroupStatusPDU	Update LS_blocks_required	LS_DATA
LS_timer expires & LS_blocks_required <> None	Send next LoadDataPDU; LS_timer := LS_T3; Update LS_blocks_required	LS_DATA
LS_timer expires & LS_blocks_required = None	Send GroupStatusRequestPDU; LS_timer := LS_T2; LS_retry_counter :=LS_retry_count	LS_WAIT
Other	Ignore	LS_DATA
CURRENT STATE: LS_WAIT		
EVENT	ACTION(S)	NEXT STATE
Receive GroupStatusPDU	Update LS_blocks_required	LS_WAIT
LS_timer expired & LS_blocks_required <> None	LS_timer := LS_T3	LS_DATA
LS_timer expired & LS_blocks_required = None & LS_retry_counter > 0	LS_timer := LS_T2; Send GroupStatusReqPDU; Decrement LS_retry_counter	LS_WAIT
LS_retry_counter = 0 & LS_timer expired & LS_blocks_required = None	Dissolve state machine	NULL
Receive acceptable LoadRequestPDU	Send LoadResponsePDU	LS_WAIT
Other	Ignore	LS_WAIT

LS_reference_id

Holds the value of the ReferenceID field sent in the LoadResponsePDU. The ReferenceID value is assigned by the LS as a unique identifier that distinguishes a particular load from any other loads that the LS is performing concurrently.

LS_block_size

Size of the load blocks. Its value shall be less than or equal to the value of BlockSize specified in the LoadRequestPDU.

LS_block_delay

Delay the LS will insert between transmission of LoadDataPDUs. The sum of LS_block_delay and the minimum amount of delay inserted by the network shall be greater than or equal to MinBlockDelay sent in the LoadResponsePDU. LS_block_delay is used to set LS_T3.

LS_address

Address used as the image destination address for a particular load operation. The address may be either a group MAC address or the individual MAC address of an LD.

LS_responses[]

A set of entries, each of which contains an address to which a LoadResponsePDU should be sent on exit from the LS_ACCUM state, along with the appropriate ExchangeID if any.

LS_number_blocks

The number of blocks in this load.

LS_number_groups

The number of Groups in this load. Except for the last Group of the load, a Group consists of 256 data blocks. This variable is calculated from: (LS_number_blocks + 255)/256.

LS_active-group_number

Holds the number of the Group being transmitted.

LS_blocks-required [LS_number_groups]

An array, each element of which contains a set of indicators that correspond to data blocks that need to be sent for a particular Group. There are a total of LS_number_groups elements and a total of LS_number_blocks indicators in the array.

LS_active_block_number

The number of the data block within the Group being sent.

LS_timer

Timer used by LS state machine. It may be set to the following intervals (these intervals differ from those used by the LD state machine):

— LS_T1 Amount of time that LS state machine will wait to accumulate LoadRequestPDUs before sending a LoadResponsePDU. If the LS state machine is dealing with a point-to-point load (i.e., the destination address for the load is an individual MAC address), the value of LS_T1 assigned to LS_timer is zero.

— LS_T2 Amount of time that the LS state machine will wait for a GroupStatusPDU after it has sent a LoadResponsePDU.
— LS_T3 Delay used to pace the transmission of data blocks. The sum of LS_T3 and the maximum amount of delay added by the network shall be less than the value of MaxBlockDelay sent in the LoadResponsePDU. The sum of LS_T3 and the minimum delay introduced by the network shall be greater than or equal to MinBlockDelay sent in the LoadResponsePDU.

NOTE—An LD will not use an LS if (LD_T1 * LD_retry_count_1) <LS_T1.

LS_retry_counter

Retry counter used for sending GroupStatusRequestPDUs to determine if any more data blocks are needed. It may be set to the value of LS_retry_count.

8.7.2.2 LS description of procedures

The SLE prepares itself to receive load-related PDUs addressed to its individual MAC address or to load server group MAC addresses and then waits for LoadRequestPDUs. On the receipt of a LoadRequestPDU, the SLE issues a SYSTEM_LOAD.indication primitive to the SLU in order to ascertain whether the LoadRequestPDU specifies an image that the SLU can provide. The SLU subsequently issues a SYSTEM_LOAD.response primitive. If the SLU cannot provide the image, the SLE ignores the LoadRequestPDU.

If the SYSTEM_LOAD.response indicates availability of the requested image, the SLE may optionally check its currently active state machines to determine if it may merge this request into a load that is in progress. A merge is possible if

— The LoadInfo field is compatible.
— The requested BlockSize is equal to or greater than LS_block_size.
— The requested MinBlockDelay is less than or equal to the LS_block_delay.
— The LoadAddress specified (if any) is the same address that is contained in LS_address.
— The calculated maximum block delay will be less than or equal to the maximum block delay requested (if specified).

If the SLE is able to merge the request,

— It returns a LoadResponsePDU to the requesting LD based upon the state variables contained in the LS state machine chosen. This is achieved

  • If the state machine is in the LS_ACCUM state, by adding the address of the LD and its ExchangeID to the set of entries held in LS_responses.
  • If the state machine is not in LS_ACCUM, by generating and sending the LoadResponsePDU immediately.

The state of the active state machine does not change in either case.

If the SLE cannot merge the request,

— It creates a new LS state machine, sets LS_timer (see note) to LS_T1, and sets the state variables LS_image_id, LS_number_blocks, and LS_number_groups based on the attributes of the load image and the fields of the LoadRequestPDU.
— An entry is created in LS_responses to record the address and exchange ID for the LoadResponsePDU.

— A unique reference ID is assigned to LS_reference_id, and LS_address is set to either a group MAC address or the individual MAC address of the LD. (The group MAC address to be used is selected from a set of group MAC addresses available for use by this LS.)
— LS_block_delay is calculated. LS_active_group_number, LS_active_block_number, and the LS_blocks_required array are cleared.
— The state machine enters the LS_ACCUM state.

NOTE—The value of LS_T1 will define how soon a LoadResponsePDU is sent. A value of 0 will cause a LoadResponsePDU to be generated immediately, whereas a value greater than 0 will allow the accumulation of multiple Load Requests prior to generation of a LoadResponsePDU.

The standard does not require that an SLE be capable of merging requests for loads, but is designed to allow merging should it be desirable for the SLE to do so.

8.7.2.2.1 LS LOAD REQUEST ACCUMULATE state (LS_ACCUM)

If an acceptable LoadRequestPDU is received, the response address and exchange ID are recorded in LS_responses.

When LS_timer expires, LoadResponsePDU(s) are sent using LS_block_delay, LS_reference_ID, LS_address, LS_number_blocks, and LS_image_id to set fields of the PDU. If multiple requests are being satisfied, a response for each request is returned, including an ExchangeID of the same value in the corresponding request if one was present. The state variable LS_response_addresses contains the set of addresses to which responses shall be sent, along with the relevant Exchange IDs.

The loadSelector may be set to encourage or discourage selection of this load by the requesting LD. A low value (typically negative) will discourage selection, while a high value (typically positive) will encourage selection. Reasons for discouraging selection include heavy loading of the LS or the allocation of the LS to the role of a backup server. The determination of the value is application-specific and not constrained by this standard.

LS_timer is then initialized to LS_T2, LS_retry_counter is set to LS_retry_count, and the state machine enters the LS_WAIT state to await the arrival of GroupStatusPDU(s).

8.7.2.2.2 LS DATA LOAD state (LS_DATA)

When the LS_timer (LS_T3) expires, a LoadDataPDU is sent. It is addressed to LS_address and contains LS_reference_ID, LS_active_group_number, LS_active_block_number, and the corresponding data block. The length of the block (carried by the LoadDataPDU) shall be equal to LS_block_size, except for the last block of the load image, which may be of a size greater than zero octets and less than or equal to LS_block_size.

After sending the LoadDataPDU, the corresponding indicator of LS_blocks_required [LS_active_group_number] is cleared. The next data block to be sent is selected by finding the next indicator that is set in LS_blocks_required. LS_active_block_number and LS_active_group_number are changed accordingly. The LS_timer is initialized using LS_T3.

If a GroupStatusPDU is received while in the LS_DATA state, indicators of LS_blocks_required [group_number] corresponding to the blocks specified in the RequiredBlocks field of the GroupStatusPDU are set, indicating that the blocks need to be retransmitted.

If LS_blocks_required is empty, LS_timer is initialized with a timeout of 0, LS_retry_counter is set to LS_retry_count and the state machine enters the LS_WAIT state. This will cause immediate generation of a GroupStatusRequestPDU.

If an acceptable LoadRequestPDU is received in the LS_DATA state, a LoadResponsePDU is sent as described under LS_ACCUM. No state change occurs as a result.

8.7.2.2.3 LS WAIT state (LS_WAIT)

While in this state, the state machine waits for GroupStatusPDUs in response to GroupStatusRequestPDUs or LoadResponsePDUs.

When a GroupStatusPDU is received while in the LS_WAIT state, then indicators in LS_blocks_required [group_number] corresponding to the blocks specified in the RequiredBlocks field of the GroupStatusPDU are set so that the indicated blocks will be retransmitted.

When LS_timer expires, if LS_blocks_required is not empty, LS_active_group_number and LS_active_block_number are set according to the first required block in LS_blocks_required. The LS_timer is initialized to LS_T3, and the state machine enters the LS_DATA state.

When LS_timer expires, if LS_blocks_required is empty, LS_retry_counter is decremented. If it has not expired, a GroupStatusRequestPDU is sent using the value of LS_reference_id, and the LS_timer initialized with LS_T2.

If an acceptable LoadRequestPDU is received in the LS_WAIT state, a LoadResponsePDU is sent as described under LS_ACCUM. No state change occurs as a result.

If LS_retry_counter expires, the state machine is dissolved.

8.7.3 Addressing of load servers

Subclause 8.7.1 describes how an LD issues a Load Request PDU addressed to the individual MAC address of an LS or to a group MAC address. The particular choice of MAC address used by an LD to locate an LS is an implementation decision; however, the IEEE 802.1 Working Group has registered a group MAC address for use as a load server address in LD or LS implementations that are conformant to this standard. The use of this address is not a conformance requirement. This address may be used by an LD as the default address for Load Request and Group Status PDUs. The address is as follows:

01-80-C2-00-00-11

NOTE—This group MAC address has been expressed using the notation described in 5.3.1 of IEEE Std 802-1990. In 5.3 of the same standard, the relationship of this representation to bit ordering in transmission is also described.

8.7.4 Addressing of loadable devices

Subclauses 8.2 and 8.7.2 describe how the address to be used by the LS for sending PDUs to an LD is determined. These PDUs may be addressed to the individual MAC address of an LD or a group MAC address. The particular choice of MAC address used by an LS to locate an LD is an implementation decision; however, the IEEE 802.1 Working Group has registered a group MAC address for use as a loadable device address in LD or LS implementations that are conformant to this standard. The use of this address is not a conformance requirement. This address may be used by an LD as the default group MAC address for reception of PDUs from an LS. The address is as follows:

01-80-C2-00-00-12

NOTE—This group MAC address has been expressed using the notation described in 5.3.1 of IEEE Std 802-1990. In 5.3 of the same standard, the relationship of this representation to bit ordering in transmission is also described.

8.8 Use of layer services

The underlying service that may be used by the SLE to convey Load PDUs between LD and LS or LS and LD is as follows:

LLC Type 1 service

The use of other services is not precluded; however, this standard does not specify any conformance-related aspects of the use of any other services.

NOTE—The operation of the System Load Protocol requires only the use of LLC Type 1 service over the LAN Management Link Service Access Point (LSAP), as described in 8.8.1. The presence of other classes of LLC service, or the support of any LLC service over other LSAPs, is not required for operation of the protocol.

8.8.1 Use of LLC Type 1 service by the SLE

The SLE generates a single DL_UNITDATA request primitive for each System Load PDU that it wishes to send. The address fields of the resultant LLC PDU shall contain the standard LSAP address reserved for use by IEEE 802.1B LAN/MAN Management (ISO/IEC DIS 15802-2). The Address Type Designator Bit of the Destination Service Access Point (DSAP) Address shall be set to “0”. The Actual Address bits of both the DSAP Address and the Source Service Access Point (SSAP) shall be encoded with the bit pattern

100 0000

where the leftmost bit is the least significant bit and the bits increase in significance from left to right. The local_address and remote_address fields in the DL_UNITDATA request primitive shall contain the address of the local SLE and remote SLE, respectively. The latter is determined as described in 8.7.

The SLE will receive DL_UNITDATA indication primitives that specify a local_address corresponding to the individual LSAP address reserved for IEEE LLC Sublayer Management and that contain valid System Load Protocol Data Units.

Use of the Priority parameter of the DL_UNITDATA service is outside the scope of this International Standard.

8.9 ASN.1 encodings

The following is an Abstract Syntax Notation One (ASN.1, specified in ISO/IEC 8824) description of the syntax of the System Load Protocol PDUs. These PDUs shall be encoded according to the Basic Encoding Rules for ASN.1, as specified in ISO/IEC 8825.

```
ieee802dot1LoadProtocol {iso(1) member-body(2) us(840) ieee802dot1partE(10010)
asn1Module(2) loadprotocol(0) version1(0)} DEFINITIONS ::=

BEGIN

IMPORTS

MACAddress

FROM IEEECommonDefinitions {iso(1) member-body(2) us(840) ieee802dot1partF(10011)
asn1Module(2) commondefinitions(0) version1(0)}

; -- End of IMPORTS

-- Define System Load Protocol PDU structures
```

```
LoadProtocol ::= loadPDU [1] LoadPDU

-- LoadPDU is a choice of NetworkManagementPDU:
-- the above construct therefore maintains compatibility with the
-- Network Management protocol. In effect, the explicit tag [1]
-- operates as a protocol identifier, distinguishing Load PDUs
-- from Network Management PDUs.

LoadPDU ::= CHOICE  {

    loadRequestPDU           [0] IMPLICIT LoadRequestPDU,
    loadResponsePDU          [1] IMPLICIT LoadResponsePDU,
    groupStatusPDU           [2] IMPLICIT GroupStatusPDU,
    groupStatusRequestPDU    [3] IMPLICIT GroupStatusRequestPDU,
    loadDataPDU              [4] IMPLICIT LoadDataPDU }

-- Choice [0] of LoadPDU

LoadRequestPDU ::= SEQUENCE  {

    exchangeID        [0] IMPLICIT ExchangeID OPTIONAL,
    loadAddress       [1] IMPLICIT LoadAddress OPTIONAL,
    size              [2] IMPLICIT BlockSize,
    minDelay          [3] IMPLICIT MinBlockDelay,
    maxDelay          [4] IMPLICIT MaxBlockDelay DEFAULT 32 767,
    reasonCode        [5] IMPLICIT LoadReason DEFAULT Unspecified,
    info              [6] IMPLICIT SEQUENCE OF LoadInfo OPTIONAL }

ExchangeID            ::= OCTETSTRING       -- 16 octets maximum

LoadAddress           ::= MACAddress        -- Defined in Common Defs

BlockSize             ::= INTEGER       -- 1..32 767

MinBlockDelay         ::= INTEGER       -- 0..32 767

MaxBlockDelay         ::= INTEGER       -- 1..32 767

LoadReason ::= INTEGER  {
    unspecified       (0),
    powerUp           (1),
    forcedLoad        (2),
    operationalFailure(3),
    loadFailure       (4),
    reconfiguration   (5) }

-- Implementation-specific shall use negative values.

LoadInfo ::= SEQUENCE  {

    privateID         [0] ANY OPTIONAL,
    stationID         [1] IMPLICIT StationID OPTIONAL,
    imageID           [2] IMPLICIT ImageID OPTIONAL }

StationID ::= SEQUENCE  {

    manufacturer      [0] IMPLICIT ManufacturerID OPTIONAL,
    device            [1] IMPLICIT DeviceTypeID OPTIONAL,
    revision          [2] IMPLICIT RevisionNumber OPTIONAL  }

ManufacturerID ::= OCTETSTRING

-- Manufacturer ID is 3 octets in length and contains an
-- organizationally unique identifier as described in
```

```
-- 5.2 of IEEE Std 802-1990. Encoding is as prescribed for the
-- corresponding octets of MACAddress as described in IEEE Std 802.1F-1993.

DeviceTypeID  ::= OCTETSTRING      -- Meaning is implementation-
                                   -- specific

RevisionNumber::= OCTETSTRING      -- Meaning is implementation-
                                   -- specific

ImageID       ::= OCTETSTRING      -- Meaning is implementation-
                                   -- specific

-- Choice [1] of LoadPDU

LoadResponsePDU ::= SEQUENCE  {

    exchangeID        [0] IMPLICIT ExchangeID OPTIONAL,
    loadAddress       [1] IMPLICIT LoadAddress,
    blockSize         [2] IMPLICIT BlockSize,
    minBlockDelay     [3] IMPLICIT MinBlockDelay,
    maxBlockDelay     [4] IMPLICIT MaxBlockDelay,
    referenceID       [5] IMPLICIT ReferenceID,
    numberBlocks      [6] IMPLICIT NumberBlocks,
    loadSelector      [7] IMPLICIT LoadSelector DEFAULT 0,
    imageInfo         [8] IMPLICIT SEQUENCE OF ImageInfo OPTIONAL }

ReferenceID   ::= OCTETSTRING      -- two octets -
                                   -- implementation-specific

NumberBlocks  ::= INTEGER          -- 0..65 535

LoadSelector  ::= INTEGER          -- -128..127

ImageInfo     ::= SEQUENCE  {

    privateID         [0] ANY OPTIONAL,
    loadInfo          [1] IMPLICIT LoadInfo OPTIONAL }

-- Choice [2] of LoadPDU

GroupStatusPDU ::= SEQUENCE  {

    referenceID       [0] IMPLICIT ReferenceID,
    groupNumber       [1] IMPLICIT GroupNumber OPTIONAL,
    requiredBlocks    [2] RequiredBlocks  }

GroupNumber ::= INTEGER -- 0..255

RequiredBlocks ::= CHOICE  {

    bitMap                  [0] IMPLICIT BitMap,
    requiredBlocksCode      [1] IMPLICIT RequiredBlocksCode }
-- If RequiredBlocksCode is the selected choice, then GroupNumber
-- is optionally present in GroupStatus. Otherwise, GroupNumber is
-- required to be present.

BitMap ::= BITSTRING        -- 256 bits max, default 0
                               -- bit N corresponds to block N of the
                               -- Group; N is in the range 1-256.

RequiredBlocksCode ::= INTEGER  {

    noBlocks   (0),
    allBlocks  (1) }
```

```
-- Choice [3] of LoadPDU

GroupStatusRequestPDU ::= SEQUENCE  {

    referenceID       [0] IMPLICIT ReferenceID  }

-- Choice [4] of LoadPDU

LoadDataPDU ::= SEQUENCE  {

    refID             [0] IMPLICIT ReferenceID,
    groupNum          [1] IMPLICIT GroupNumber,
    blockNum          [2] IMPLICIT Block,
    dataBlock         [3] LoadData  }

Block     ::= INTEGER   -- 0..255

LoadData  ::= ANY

END -- End of Load Protocol Definitions
```

Table 3 shows the correspondence between the field identifiers used in the ASN.1 definition of the PDU structures and the PDU field names used in the remainder of the text.

Table 3—PDU field name/ASN.1 field identifier correspondence

PDU	PDU field name	ASN.1 field identifier
LoadRequestPDU	ExchangeID	exchangeID
"	LoadAddress	loadAddress
"	BlockSize	size
"	MinBlockDelay	minDelay
"	MaxBlockDelay	maxDelay
"	LoadReason	reasonCode
"	LoadInfo	info
LoadResponsePDU	ExchangeID	exchangeID
"	LoadAddress	loadAddress
"	BlockSize	blockSize
"	MinBlockDelay	minBlockDelay
"	MaxBlockDelay	maxBlockDelay
"	ReferenceID	referenceID
"	NumberBlocks	numberBlocks
"	LoadSelector	loadSelector
"	ImageInfo	imageInfo
GroupStatusPDU	ReferenceID	referenceID
"	GroupNumber	groupNumber
"	RequiredBlocks	requiredBlocks
GroupStatusRequestPDU	ReferenceID	referenceID
LoadDataPDU	ReferenceID	refID
"	GroupNumber	groupNum
"	Block	blockNum
"	LoadData	dataBlock

9. System Load Protocol managed object classes

9.1 Introduction

Subclause 9.2 specifies the load-related managed objects, their attributes, and the management operations associated with them that constitute the functionality of the SL_MIB. Protocol definitions related to manipulation of these managed objects, when accessed via the LAN/MAN Management Protocol (defined in IEEE 802.1B LAN/MAN Management [ISO/IEC DIS 15802-2]), appear in 9.3.

9.1.1 Approach

Clause 9 specifies aspects of the operation of the System Load Protocol that

a) Characterize local interactions between the protocol machine and management.
b) Reflect and support remote exchanges of information pertaining to the protocol machine and operations on that information for management purposes using management protocols.
c) Define the concrete encodings that are used to communicate such management information, in particular management protocols.

Implementations that do not provide remote access to network management are not required to conform to this subclause, since the management of a station by entirely local means is outside the scope of this International Standard.

This clause is organized into subclauses that contain the following:

— Definitions of the managed objects that pertain to this standard. The definitions include definitions of any contained managed objects or attributes, and definitions of operations that may be performed upon them. These managed objects are defined in a manner that is independent of the management protocol that will be used to invoke operations upon them, but it is intended that the definitions will be compatible with use of the LAN/MAN Management Protocol (defined in IEEE 802.1B LAN/MAN Management [ISO/IEC DIS 15802-2]) and CMIS/CMIP (defined in ISO/IEC 9595 and 9596).
— Definitions of any protocol-related encodings or structures that are required in order to specify the use of particular management protocols.

9.1.2 Summary of facilities

Management facilities are specified in this standard for the following purposes:

a) Establishment and manipulation of counter, delay, and timer values associated with the operation of the LD or LS state machine.
b) Establishment and manipulation of state information associated with the operation of the LD or LS state machine.
c) Third party initiation of a Load.
d) Announcement of active LS's.

9.1.3 Relationship to specific management functional areas

The management facilities described in this standard relate to the Configuration Management functional area, as described in ISO 7498-4.

9.1.4 Definition of terms

The specification of the management facilities that relate to the System Load Protocol makes use of terminology that is specified or referenced in IEEE Std 802.1F-1993. The notation used for the definition of managed object classes in this standard is defined in ISO/IEC 10165-4.

9.1.5 Conformance

Conformance to the management of the System Load Protocol requires the following:

a) The support of one or both of the LS and LD managed objects and all the operations and events that are associated with them.
b) The support of the Resource Type ID managed object class.
c) The support of remote access to the managed objects by means of the 802.1B LAN/MAN Management protocol (ISO/IEC DIS 15802-2) or CMIP (ISO/IEC 9596).

9.2 System Load Protocol managed object definitions

There are two managed objects associated with the operation of the System Load Protocol:

a) The LD managed object
b) The LS managed object

They relate to the operation of the LD and LS state machines, respectively. In addition, the Resource Type managed object is present if management is to be performed using the LAN/MAN Management Protocol defined in IEEE 802.1B LAN/MAN Management (ISO/IEC DIS 15802-2); its definition appears in the ASN.1 specifications for this protocol (see 9.3.2).

9.2.1 LD Managed object definition

The LD managed object contains attribute values that hold timer, counter, and status information related to the operation of the LD.

9.2.1.1 Definitions of contained managed objects

Each instance of the LD managed object class contains a single instance of the Resource Type ID managed object class.

9.2.1.2 Definitions of contained attributes

The LD managed object contains the following attributes:

LD Name

LD Name contains the name of the LD managed object. Its value is read only.

LD Load Info

LD Load Info contains fields that describe the equipment requesting the load and the load image needed. The value of LD Load Info is used in the LoadInfo field of LoadRequestPDUs issued by the system. The following fields may optionally appear in Load Info:

a) A Station Identifier field, containing the optional values as octet strings:
   Manufacturer ID,
   Device Type ID,
   Revision.
b) An Image Identifier field, containing an octetstring that specifies which image to load.
c) Unspecified, privately defined information.

LD Load Server Address

This is the LS MAC address to which the LD shall transmit LoadRequestPDUs.

LD T1

LD T1 is the amount of time in milliseconds that the LD shall wait for a LoadResponsePDU before issuing or reissuing a LoadRequestPDU. The LD state variable LD_timer is set to the value of this parameter as specified in 8.7.1.1.

LD T2

LD T2 is the amount of time in milliseconds that the LD shall wait for a LoadDataPDU (or a GroupStatusRequestPDU) before sending a GroupStatusPDU. The LD state variable LD_timer is set to the value of this parameter as specified in 8.7.1.1.

LD Retry Count 1

LD Retry Count 1 is a count of the number of times a LoadRequestPDU shall be sent before LD_FAILED state is entered. The LD state variable LD_retry_ counter is set to this value as specified in 8.7.1.1.

LD Retry Count 2

LD Retry Count 2 is a count of the number of times the LD state machine shall wait for load-related PDUs from the LS. The LD state variable LD_retry counter is set to the value of this parameter as specified in 8.7.1.1.

LD Block Size

LD Block Size is used to set the BlockSize field in the LoadRequestPDU. It is the maximum block size in octets that the LD can accept.

LD Min Block Delay

LD Min Block Delay is used to set the MinBlockDelay field in the LoadRequestPDU. It is the minimum amount of delay in milliseconds that the LD requires between transmission of data blocks.

LD Max Block Delay

LD Max Block Delay is used to set the MaxBlockDelay field in the LoadRequestPDU. It is the maximum amount of delay in milliseconds that the LD will expect between transmission of data blocks.

LD Status

LD Status indicates whether or not the LD may request a load. If set to Disabled, the LD may not request a load; if set to Enabled, the LD may request a load.

9.2.1.3 Definitions of operations and events

The following operations may be performed upon the LD managed object:

Read LD attribute

Purpose: To obtain the value of one of the attributes of the LD object. All LD attributes may be read.

Inputs: Attribute identifier.

Outputs: Attribute value.

Modify LD attribute

Purpose: To modify the value of one of the attributes of the LD managed object. All LD attributes may be modified, with the exception of the LD Name attribute, which is read only.

Inputs: Attribute identifier, desired new value.

Outputs: None.

Load

Purpose: The Load operation requests that the LD initiate a load.

Inputs: A value of LoadInfo may be provided.

Outputs: The Load operation returns the status of the requested load operation. The possible status values are the following:

a) WillComply: Will attempt to initiate a load using the LoadInfo (if provided).
b) RefuseToComply: Will not attempt to load for unspecified reason.
c) InvalidLoadInfo: Will not attempt to load due to invalid LoadInfo.
d) InvalidState: Will not attempt to load due to LD not being in a state to initiate a load.

The LD managed object may generate the following events: None.

9.2.1.4 Definition of error handling

There are no special error handling requirements associated with the LD managed object.

9.2.1.5 Definitions of other relationships

No further relationships exist between the LD managed object and other managed objects.

9.2.2 LS managed object definition

The LS managed object contains attribute values that hold timer, counter, and status information related to the operation of the LS.

9.2.2.1 Definitions of contained managed objects

Each instance of the LS managed object class contains a single instance of the Resource Type ID managed object class.

9.2.2.2 Definitions of contained attributes

The LS managed object contains the following attributes:

LS Name

LS Name contains the name of the LS managed object. Its value is read only.

LS T1

LS T1 is the amount of time in milliseconds an instance of an LS state machine shall wait in LS_ACCUM state before sending a LoadResponsePDU. The LS state variable LS_timer is set to the value of this parameter as specified in 8.7.2.1.

LS T2

LS T2 is the amount of time in milliseconds an instance of an LS state machine shall wait for a GroupStatusPDU after it has sent a LoadResponsePDU or a GroupStatusRequestPDU. The LS state variable LS_timer is set to the value of this parameter as specified in 8.7.2.1.

LS Net Delay

LS Net Delay is the expected time in milliseconds that the LS shall assume for network delay to the LD. This time is added to the minimum block delay and the expected internal LS delay to determine the maximum block delay.

LS Retry Count

LS Retry Count is the number of times an instance of an LS state machine shall send GroupStatusRequestPDUs.

LS Status

LS Status indicates whether the LS is disabled (i.e., cannot perform loads) or enabled (i.e., can perform loads).

9.2.2.3 Definitions of operations and events

The following operations may be performed upon the LS managed object:

Read LS attribute

Purpose: To obtain the value of one of the attributes of the LS object. All LS attributes may be read.

Inputs: Attribute identifier.

Outputs: Attribute value.

Modify LS attribute

Purpose: To modify the value of one of the attributes of the LS managed object. All LS attributes may be modified, with the exception of the LS Name attribute, which is read only.

Inputs: Attribute identifier, desired new value.

Outputs: None.

The LS managed object may generate the following events:

LS Active

Purpose: The LS Active event indicates that an LS has become available to service requests at the MAC address specified in LS address. The LS may optionally send the LSActiveEvent PDU at intervals to allow LDs that become active after the LS first became active to be made aware of the availability of the LS.

Outputs: The information carried in the LS Active event is as follows:

Private LS Info

Private LS Info specifies implementation-specific information, if present.

LS Address

LS Address specifies a MAC address to which LoadRequestPDUs may be sent.

9.2.2.4 Definition of error handling

There are no special error handling requirements associated with the LD managed object.

9.2.2.5 Definitions of other relationships

No further relationships exist between the LS managed object and other managed objects.

9.3 System Load Protocol managed object class definitions

The following subclauses define the managed object classes described in 9.2 using the template notation defined in ISO/IEC 10165-4. Further information on the use of this notation in the context of the IEEE 802.1B LAN/MAN Management protocol can be found in IEEE Std 802.1F-1993.

9.3.1 LD managed object class definition

The LD managed object class provides the capability to remotely manage aspects of the operation of the LD, as described in 9.2.1. The definition contains a mandatory package, which includes the naming attribute and the Load action, and a conditional package, which includes the remaining attributes.

Support for this managed object class is optional. If the managed object class is supported, support of the attributes package is optional.

```
oLD  MANAGED OBJECT CLASS
   DERIVED FROM  "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2 : 1992":top;
   CHARACTERIZED BY
        pLD1          PACKAGE
          ATTRIBUTES  aLDName   GET;   -- Naming Attribute
          ACTIONS     acLoad;
        ;
   ;
   CONDITIONAL PACKAGES
        pLD2          PACKAGE
          ATTRIBUTES  aLDLoadInfo               GET-REPLACE,
                      aLDLoadServerAddress      GET-REPLACE,
                      aLDT1                     GET-REPLACE,
                      aLDT2                     GET-REPLACE,
```

```
                        aLDRetryCount1              GET-REPLACE,
                        aLDRetryCount2              GET-REPLACE,
                        aLDBlockSize                GET-REPLACE,
                        aLDMinBlockDelay            GET-REPLACE,
                        aLDMaxBlockDelay            GET-REPLACE,
                        aLDStatus                   GET-REPLACE;
        REGISTERED AS       {iso(1) member-body(2) us(840) ieee802dot1partE(10010) package
                            (4) pld2(0)};
    PRESENT IF    !The implementation supports the manipulation of the attributes that the
                  package contains.!
    ;
REGISTERED AS     {iso(1) member-body(2) us(840) ieee802dot1partE(10010)
                  managedObjectClass(3) ldclass(0)};

nbLDBinding             NAME BINDING
    SUBORDINATE OBJECT CLASS                        oLD AND SUBCLASSES;
    NAMED BY SUPERIOR OBJECT CLASS
           "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2 : 1992":system AND SUBCLASSES;
    WITH ATTRIBUTE                                  aLDName;
    BEHAVIOUR
         bLDBinding     BEHAVIOUR
           DEFINED AS   !A single instance of the LD managed object class exists within the
                        superior object class.  It cannot be created or deleted dynamically
                        by management action!;
         ;
REGISTERED AS     {iso(1) member-body(2) us(840) ieee802dot1partE(10010) nameBinding(6)
                  ldbinding(0)};
```

9.3.1.1 LDName attribute

```
aLDName                 ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   ieee802dot1-LoadDefinitions.LDName;
    MATCHES FOR         EQUALITY;
    BEHAVIOUR
         bLDName        BEHAVIOUR
           DEFINED AS   !This attribute is used to name the instance of the LD managed
                        object within the systems managed object.  The value of this name
                        attribute is fixed and is equal to the string "LD".!;
    ;
REGISTERED AS     {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                  ldname(0)};
```

9.3.1.2 LDLoadInfo attribute

```
aLDLoadInfo ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   ieee802dot1-LoadDefinitions.LoadInfo;
    MATCHES FOR         EQUALITY;
    BEHAVIOUR
         bLoadInfo      BEHAVIOUR
           DEFINED AS   !The behaviour of this attribute is defined in 9.2.1.2.!;
    ;
REGISTERED AS     {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                  loadinfo(1)};
```

9.3.1.3 LDLoadServerAddress attribute

```
aLDLoadServerAddress ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   ieee802dot1-LoadDefinitions.MACAddress;
    MATCHES FOR         EQUALITY;
    BEHAVIOUR
         bLDLoadServerAddress  BEHAVIOUR
           DEFINED AS   !The behaviour of this attribute is defined in 9.2.1.2.!;
    ;
REGISTERED AS     {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                  ldloadserveraddress(2)};
```

9.3.1.4 LDT1 attribute

```
aLDT1 ATTRIBUTE
   DERIVED FROM       "CCITT Rec. X.723 | ISO/IEC 10165-5":timer;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDT1         BEHAVIOUR
          DEFINED AS  !The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldt1(3)};
```

9.3.1.5 LDT2 attribute

```
aLDT2 ATTRIBUTE
   DERIVED FROM       "CCITT Rec. X.723 | ISO/IEC 10165-5":timer;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDT2         BEHAVIOUR
          DEFINED AS  !The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldt2(4)};
```

9.3.1.6 LDRetryCount1 attribute

```
aLDRetryCount1 ATTRIBUTE
   WITH ATTRIBUTE SYNTAX   ieee802dot1-LoadDefinitions.RetryCount;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDRetryCount1  BEHAVIOUR
          DEFINED AS     The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldretrycount1(5)};
```

9.3.1.7 LDRetryCount2 attribute

```
aLDRetryCount2 ATTRIBUTE
   WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.RetryCount;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDRetryCount2    BEHAVIOUR
          DEFINED AS    !The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldretrycount2(6)};
```

9.3.1.8 LDBlockSize attribute

```
aLDBlockSize ATTRIBUTE
   WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.BlockSize;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDBlockSize      BEHAVIOUR
          DEFINED AS  !The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldblocksize(7)};
```

9.3.1.9 LDMinBlockDelay attribute

```
aLDMinBlockDelay ATTRIBUTE
   WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.BlockDelay;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDMinBlockDelay  BEHAVIOUR
          DEFINED AS  The behaviour of this attribute is defined in 9.2.1.2.!;
```

```
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldminblockdelay(8)};
```

9.3.1.10 LDMaxBlockDelay attribute

```
aLDMaxBlockDelay ATTRIBUTE
   WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.BlockDelay;
   MATCHES FOR        EQUALITY, ORDERING;
   BEHAVIOUR
        bLDMaxBlockDelay  BEHAVIOUR
          DEFINED AS   The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldmaxblockdelay(9)};
```

9.3.1.11 LDStatus attribute

```
aLDStatus ATTRIBUTE
   WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.Status;
   MATCHES FOR        EQUALITY;
   BEHAVIOUR
        bLDStatus      BEHAVIOUR
          DEFINED AS   !The behaviour of this attribute is defined in 9.2.1.2.!;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                ldstatus(10)};
```

9.3.1.12 Load action

```
acLoad ACTION
   BEHAVIOUR
        bLoad          BEHAVIOUR
          DEFINED AS   !The behaviour of this action is defined in 9.2.1.3.!;
   ;
   MODE CONFIRMED;
   WITH INFORMATION SYNTAX           ieee802dot1-LoadDefinitions.LoadInfo;
   WITH REPLY SYNTAX                 ieee802dot1-LoadDefinitions.LoadStatus;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010) action(9) load(0)};
```

9.3.2 LS managed object class definition

The LS managed object class provides the capability to remotely manage aspects of the operation of the LS, as described in 9.2.2.

Support for this managed object class is optional.

```
oLS  MANAGED OBJECT CLASS
   DERIVED FROM       "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2 : 1992":top;
   CHARACTERIZED BY
        pLS  PACKAGE
          ATTRIBUTES   aLSName        GET,-- Naming Attribute
                       aLST1          GET-REPLACE,
                       aLST2          GET-REPLACE,
                       aLSNetDelay    GET-REPLACE,
                       aLSRetryCount  GET-REPLACE,
                       aLSStatus      GET-REPLACE;
          NOTIFICATIONS nLSActive;
        ;
   ;
REGISTERED AS   {iso(1) member-body(2) us(840) ieee802dot1partE(10010)
                managedObjectClass(3) lsclass(1)};
nbLSBinding     NAME BINDING
   SUBORDINATE OBJECT CLASS          oLS AND SUBCLASSES;
   NAMED BY SUPERIOR OBJECT CLASS
        "CCITT Rec. X.721 (1992) | ISO/IEC 10165-2 : 1992":system AND SUBCLASSES;
   WITH ATTRIBUTE     aLSName;
```

```
    BEHAVIOUR
         bLSBinding     BEHAVIOUR
           DEFINED AS   !A single instance of the LS managed object class exists within the
                        superior object class.  It cannot be created or deleted dynamically
                        by management action!;
    ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) nameBinding(6)
                 lsbinding(1)};
```

9.3.2.1 LSName attribute

```
aLSName  ATTRIBUTE
    WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.LSName;
         MATCHES FOR    EQUALITY;
         BEHAVIOUR
           bLSName      BEHAVIOUR
           DEFINED AS   !This attribute is used to name the instance of the LS managed
                        object within the systems managed object.  The value of this name
                        attribute is fixed and is equal to the string "LS".!;
         ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                 lsname(11)};
```

9.3.2.2 LST1 attribute

```
aLST1 ATTRIBUTE
    DERIVED FROM  "CCITT Rec. X.723 | ISO/IEC 10165-5":timer;
    MATCHES FOR   EQUALITY, ORDERING;
    BEHAVIOUR
         bLST1          BEHAVIOUR
           DEFINED AS   !The behaviour of this attribute is defined in 9.2.2.2.!;
    ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                 lst1(12)};
```

9.3.2.3 LST2 attribute

```
aLST2 ATTRIBUTE
    DERIVED FROM  "CCITT Rec. X.723 | ISO/IEC 10165-5":timer;
    MATCHES FOR   EQUALITY, ORDERING;
    BEHAVIOUR
         bLST2          BEHAVIOUR
           DEFINED AS   !The behaviour of this attribute is defined in 9.2.2.2.!;
    ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                 lst2(13)};
```

9.3.2.4 LSNetDelay attribute

```
aLSNetDelay ATTRIBUTE
    WITH ATTRIBUTE SYNTAX   ieee802dot1-LoadDefinitions.NetDelay;
    MATCHES FOR   EQUALITY, ORDERING;
    BEHAVIOUR
         bLSNetDelay    BEHAVIOUR
           DEFINED AS   !The behaviour of this attribute is defined in 9.2.2.2.!;
    ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                 lsnetdelay(14)};
```

9.3.2.5 LSRetryCount attribute

```
aLSRetryCount ATTRIBUTE
    WITH ATTRIBUTE SYNTAX  ieee802dot1-LoadDefinitions.RetryCount;
    MATCHES FOR   EQUALITY, ORDERING;
    BEHAVIOUR
         bLSRetryCount  BEHAVIOUR
           DEFINED AS   !The behaviour of this attribute is defined in 9.2.2.2.!;
    ;
```

```
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                 lsretrycount(15)};
```

9.3.2.6 LSStatus attribute

```
aLSStatus ATTRIBUTE
   WITH ATTRIBUTE SYNTAXieee802dot1part-LoadDefinitions.Status;
   MATCHES FOR EQUALITY;
   BEHAVIOUR
        bLSStatus      BEHAVIOUR
          DEFINED AS   !The behaviour of this attribute is defined in 9.2.2.2.!;
   ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)
                 lsstatus(16)};
```

9.3.2.7 LSActive notification

```
nLSActive NOTIFICATION
   BEHAVIOUR
        bLSActive      BEHAVIOUR
          DEFINED AS   !The behaviour of this notification is defined in 9.2.2.3.!;
   ;
   WITH INFORMATION SYNTAXieee802dot1part-LoadDefinitions.LSEventInfo;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) notification(10)
                 lsactive(0)};
```

9.3.3 Resource Type ID managed object class

A single instance of the Resource Type ID managed object class is contained in each instance of the LD and LS managed object classes. The managed object class definition itself appears in IEEE Std 802.1F-1993, therefore only the name bindings appear in this International Standard; however, conformance with the definition contained in IEEE Std 802.1F-1993 is a requirement of this standard. The Resource Type ID managed object class contains manufacturer and product information related to the implementation of the LD or LS functionality.

Support for this managed object class is mandatory if support for either the LD or LS managed object classes is claimed.

```
nbResourceTypeID-LS  NAME BINDING
   SUBORDINATE OBJECT CLASS          "IEEE Std 802.1F-1993":oResourceTypeID AND
                                     SUBCLASSES;
   NAMED BY SUPERIOR OBJECT CLASS    oLS AND SUBCLASSES;
   WITH ATTRIBUTE                    "IEEE Std 802.1F-1993":aResourceTypeIDName;
   BEHAVIOUR
        bResourceTypeID-LS         BEHAVIOUR
          DEFINED AS     !A single instance of the Resource Type ID managed object class
                         exists within each instance of the superior object class.  It
                         cannot be created or deleted dynamically by management action!;
        ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) nameBinding(6)
                 resourcetypeID-lsbinding(2)};

nbResourceTypeID-LD  NAME BINDING
   SUBORDINATE OBJECT CLASS          "IEEE Std 802.1F-1993":oResourceTypeID AND
                                     SUBCLASSES;
   NAMED BY SUPERIOR OBJECT CLASS    oLD AND SUBCLASSES;
   WITH ATTRIBUTE                    "IEEE Std 802.1F-1993":aResourceTypeIDName;
   BEHAVIOUR
        bResourceTypeID-LD         BEHAVIOUR
          DEFINED AS     !A single instance of the Resource Type ID managed object class
                         exists within each instance of the superior object class.  It
                         cannot be created or deleted dynamically by management action!;
        ;
REGISTERED AS    {iso(1) member-body(2) us(840) ieee802dot1partE(10010) nameBinding(6)
                 resourcetypeID-ldbinding(3)};
```

9.3.4 Supporting ASN.1 definitions

```
ieee802dot1-LoadDefinitions {iso(1) member-body(2) us(840) ieee802dot1partE(10010)
asn1Module(2) loaddefinitions(1) version1(0)} DEFINITIONS ::=

BEGIN

IMPORTS

ManagementExtension

FROM Attribute.ASN1Module {joint-iso-ccitt ms(9) smi(3) part2(2) asn1Module(2) 1}

ManufacturerID, DeviceTypeID, RevisionNumber, ImageID

FROM ieee802dot1LoadProtocol {iso(1) member-body(2) us(840) ieee802dot1partE(10010)
asn1Module(2) loadprotocol(0) version1(0)}

MACAddress

FROM IEEECommonDefinitions {iso(1) member-body(2) us(840) ieee802dot1partF(10011)
asn1Module(2) commondefinitions(0) version1(0)}

; -- End of IMPORTS
LDName ::= GraphicString "LD"
LSName ::= GraphicString "LS"
RetryCount ::= INTEGER (0..127)
BlockSize ::= INTEGER (0..32767)
BlockDelay ::= INTEGER (0..32767)
NetDelay ::= INTEGER (0..32767)
Status ::= INTEGER {
        (0) Enabled,
        (255) Disabled }
LoadStatus ::= INTEGER {
        WillComply       (0),
        RefuseToComply   (1),
        InvalidLoadInfo  (2),
        InvalidState     (3) }
LSEventInfo ::= SEQUENCE {
   address      [0] IMPLICIT MACAddress
   extension    [1] SET OF ManagementExtension OPTIONAL }
LoadInfo ::= SEQUENCE {
   stationId    [0] IMPLICIT StationID OPTIONAL,
   imageId      [1] IMPLICIT ImageID OPTIONAL,
   extension    [2] SET OF ManagementExtension OPTIONAL }
StationID ::= SEQUENCE {
   manufacturer [0] IMPLICIT ManufacturerID OPTIONAL,
   device       [1] IMPLICIT DeviceTypeID OPTIONAL,
   revision     [2] IMPLICIT RevisionNumber OPTIONAL }
END
```

9.3.5 Mappings between LAN/MAN Management services and operations on managed objects

The following table shows the mappings of the management operations and events specified in 9.2.1.3 and 9.2.2.3 onto the LMMS service elements provided by IEEE 802.1B, LAN/MAN Management (ISO/IEC DIS 15802-2).

Table 4—Mappings of management operations

Management operation	LMMS service element(s)	Managed object
Read LD attribute	M-GET, M-CANCEL-GET	LD managed object
Modify LD attribute	M-SET	LD managed object
Load	M-ACTION	LD managed object
Read LS attribute	M-GET, M-CANCEL-GET	LS managed object
Modify LS attribute	M-SET	LS managed object
LS active	M-EVENT-REPORT	LS managed object

10. Conformance

10.1 Conformance to this International Standard

An implementation claiming conformance to this International Standard shall exhibit an external behavior consistent with having implemented the state tables and protocol elements associated with either the LS or the LD or both.

An implementation claiming conformance to the management aspects of this standard shall exhibit external behavior consistent with having implemented those elements defined in the System Load Layer Management Entity Definitions that are mandatory for the LS or the LD, or both. The mandatory elements are specified in 9.1.5.

10.2 Claims of conformance

Claims of conformance shall state the following:

a) Which functional elements have been implemented.
b) Which optional functions have been implemented.

The supplier of a protocol implementation that is claimed to conform to ISO/IEC 15802-4 : 1994 shall complete a copy of the PICS proforma provided in annex A and shall provide the information necessary to identify both the supplier and the implementation.

Annexes

Annex A[1]
PICS proforma

(normative)

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to this International Standard shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use

a) By the protocol implementor, as a checklist to reduce the risk of failure to conform to the standard through oversight.
b) By the supplier and acquirer, or potential acquirer, of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma.
c) By the user, or potential user, of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from an incompatible PICS).
d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A.2 Abbreviations and special symbols

M	Mandatory
O	Optional
O.1	Optional, but support of at least one of the group of options labelled in this way is required.
<item>:	conditional-item symbol, dependent upon the support marked for <item>: (see A.3.4).
N/A	Not Applicable

A.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

The first part of the PICS proforma, Identification, A.4, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire divided into four subclauses containing groups of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No), or by entering a value or a set or range of values.

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

Each item is identified by an item reference in the first column; the second column contains the question to be answered; and the third column contains the reference or references to the material that specifies the item in the main body of this International Standard. The remaining columns record the status of the item—whether support is mandatory or optional—and provide the space for answers (see also A.3.4).

A supplier may also provide, or can be required to provide, further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled A*i* or X*i*, respectively, for cross-referencing purposes, where *i* is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformance Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, in case that makes for easier and clearer presentation of the information.

A.3.2 Additional Information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. An example might be an outline of the ways in which a single implementation can be set up to operate in a variety of environments and configurations.

References to items of Additional Information may be entered next to any answers in the questionnaire, and may be included in items of Exception Information.

A.3.3 Exception Information

It may occasionally happen that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this; instead, the supplier shall write the missing answer in the Support column, together with an X*i* reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to ISO/IEC 15802-4 : 1994.

NOTE—A possible reason for the situation described above is that a defect in this standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional items

The PICS proforma contains a number of conditional items. These are items for which the applicability of the item itself, and its status if it does apply—mandatory or optional—are dependent upon whether or not certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the "Not Applicable" answer is marked. Otherwise, individual conditional items are indicated by a conditional symbol of the form "<item>: S" where <item> is an item reference that appears in the first column of the table for some other item, and S is one of the status symbols M or O.

If the item referred to by the conditional symbol is marked as supported, the conditional item is applicable and its status is given by S; the support column is to be completed in the usual way. Otherwise, the conditional item is not relevant and the Not Applicable (N/A) answer is to be marked.

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

A.4 Identification

A.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification—e.g., name(s) and version(s) for machine(s) and/or operating systems; System Name(s)	

NOTES

1—Only the first three items are required for all implementations; the other information may be completed as appropriate in meeting the requirements for full identification.

2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

A.4.2 Protocol summary, ISO/IEC 15802-4 : 1994 convergence protocol

Identification of protocol specification	ISO/IEC 15802-4 : 1994
Identification of amendments and corrigenda to this PICS proforma which have been completed as part of this PICS	ISO/IEC 15802-4 : 1994/ Amd. : Corr. : Amd. : Corr. :
Have any Exception items been required (see A.3.3)? (The answer Yes means that the implementation does not conform to ISO/IEC 15802-4 : 1994.)	No [ ] Yes [ ]
Date of Statement	

A.5 Major capabilities

Item	Feature	References	Status	Support
	Does the implementation support the functionality described for:			
*SLls	— Operation of the Load Server?	8.7.2, 10.1	O.1	Yes [ ] No [ ]
*SLld	— Operation of the Loadable Device?	8.7.1, 10.1	O.1	Yes [ ] No [ ]
*MOls	— Management of the Load Server?	9.1.5, 9.2.2, 10.1	SLls:O	Yes [ ] No [ ] N/A [ ]
*MOld	— Management of the Loadable Device?	9.1.5, 9.2.1, 10.1	SLld:O	Yes [ ] No [ ] N/A [ ]

A.6 PDU support

A.6.1 Loadable Device support

If operation of the Loadable Device is not supported, item SLld, mark N/A and continue at A.6.2. **N/A [ ]**

Item	Feature	References	Status	Support
LDlrqpdu	Does the LD support the transmission of LoadRequest PDUs?	8.2	M	Yes [ ]
LDlrspdu	Does the LD support the reception of LoadResponse PDUs?	8.3	M	Yes [ ]
LDgspdu	Does the LD support the transmission of GroupStatus PDUs?	8.4	M	Yes [ ]
LDgsrpdu	Does the LD support the reception of GroupStatusRequest PDUs?	8.5	M	Yes [ ]
LDldpdu	Does the LD support the reception of LoadData PDUs?	8.6	M	Yes [ ]
LDlsact	Does the LD support the reception of LS Active event report PDUs?	9.2.2.3, 9.3.2.7	O	Yes [ ] No [ ]
LDload	Does the LD support the reception of Load action PDUs?	9.2.1.3, 9.3.1.12	MOld:M	Yes [ ] N/A [ ]

A.6.2 Load Server support

If operation of the Load Server is not supported, item SLls, mark N/A and continue at A.7. **N/A [ ]**

Item	Feature	References	Status	Support
LSlrqpdu	Does the LS support the reception of LoadRequest PDUs?	8.2	M	Yes [ ]
LSlrspdu	Does the LS support the transmission of LoadResponse PDUs?	8.3	M	Yes [ ]
LSgspdu	Does the LS support the reception of GroupStatus PDUs?	8.4	M	Yes [ ]
LSgsrpdu	Does the LS support the transmission of GroupStatusRequest PDUs?	8.5	M	Yes [ ]
LSldpdu	Does the LS support the transmission of LoadData PDUs?	8.6	M	Yes [ ]
LSlsact	Does the LS support the transmission of LS Active event report PDUs?	9.2.2.3, 9.3.2.7	MOls:M	Yes [ ] N/A [ ]
LSload	Does the LS support the transmission of Load action PDUs?	9.2.1.3, 9.3.1.12	O	Yes [ ] No [ ]

A.7 Support of PDU parameters and protocol features

A.7.1 Loadable Device support

If operation of the Loadable Device is not supported, item SLld, mark N/A and continue at A.7.2. **N/A [ ]**

Item	Feature	References	Status	Support
*LDmor	Does the LD support multiple outstanding load requests? In transmitted LoadRequest PDUs:	8.2.2	O	Yes [ ] No [ ]
LDlreid	—Does the LD support use of the ExchangeID field?	8.2.2	LDmor:M	Yes [ ] N/A [ ]
*LDlrlaf	—Does the LD support use of the LoadAddress field?	8.2.2	O	Yes [ ] No [ ]
LDlriad	—Can the individual address of the station be used in the LoadAddress field?	8.2.2	LDlrlaf:O	Yes [ ] No [ ] N/A [ ]
LDlrgad	—What group MAC addresses may be used in the LoadAddress field?	8.2.2	LDlrlaf:O	Values_____ N/A [ ]
LDlrdef	—By default (i.e., if no LoadAddress is specified in a request) can the LD accept a load addressed to its individual MAC address or to any group address at the option of the LS?	8.2.2	M	Yes [ ]

Item	Feature	References	Status	Support
LDlrbs	—Is the range of values for BlockSize supported by the LD in the range 1–32 767?	8.2.2	M	From__ to__
LDlrmbd	—Is the value of MinBlockDelay supported by the LD in the range 0–32 767?	8.2.2	M	Value___
LDlrmxd	—Is the value of MaxBlockDelay supported by the LD in the range MinBlockDelay –32 767?	8.2.2	M	Value___
LDlrlres	—What values does the LD support for the LoadReason field?	8.2.2	O	Values____
LDlrlinf	—What values does the LD support for the LoadInfo field?	8.2.2	O	Values____
	In transmitted GroupStatus PDUs:			
LDgsref	—Does the LD support the use of the ReferenceID field?	8.4.2	M	Yes [ ]
LDgsgn	—Is the range of values supported for GroupNumber in the range 0–255?	8.4.2	M	From__ to__
LDgsrb	—Are both the BitMap and RequiredBlocksCode variants of RequiredBlocks supported?	8.4.2	M	Yes [ ]
	In received LoadResponsePDUs:			
LDlrseid	—Does the LD support the use of the ExchangeID field?	8.2.2, 8.3.2	LDmor:M	Yes [ ] N/A [ ]
LDlrsla	—Does the LD support the use of both individual and group addresses as the destination address for a load?	8.2.2, 8.3.2	O	Yes [ ] No [ ]
LDlrsbs	—Is the range of values for Block-Size supported by the LD in the range 1–32 767?	8.3.2	M	From__ to__
LDlrsmbd	—Is the value of MinBlockDelay supported by the LD in the range 1–32 767?	8.2.2, 8.3.2	M	Value__
LDlrsmxd	—Is the value of MaxBlockDelay supported by the LD in the range 1–32 767?	8.2.2, 8.3.2	M	Value__
LDlrsrid	—Does the LD support the use of the ReferenceID field?	8.3.2	M	Yes [ ]
LDlrsnb	—Is the range of values of NumberBlocks supported by the LD in the range 1–65 535?	8.3.2	M	From__ to__
LDlrsls	—Does the LD support use of the LoadSelector field?	8.3.2, 8.3.3	M	Yes [ ]
LDlrsii	—Does the LD support use of the ImageInfo field?	8.3.2	M	Yes [ ]

Item	Feature	References	Status	Support
	In received GroupStatusRequestPDUs:			
LDgsrda	—Does the LD support the use of both individual and group MAC addresses as the destination address?	8.5.1	O	Yes [ ] No [ ]
LDgsrrid	—Does the LD support the use of the ReferenceID field?	8.5.2	M	Yes [ ]
	In received LoadDataPDUs:			
LDlorid	—Does the LD support the use of the ReferenceID field?	8.6.2	M	Yes [ ]
LDlogn	—Is the range of values supported for GroupNumber in the range 0–255?	8.6.2	M	From__ to__
LDlobl	—Is the range of values supported for Block in the range 0–255?	8.6.2	M	From__ to__
LDlold	—Does the LD support the use of the LoadData field?	8.6.2	M	Yes [ ]

A.7.2 Load Server support

If operation of the Load Server is not supported, item SLls, mark N/A and continue at A.7. **N/A [ ]**

Item	Feature	References	Status	Support
	In received LoadRequest PDUs:			
LSlreid	—Does the LS support use of the ExchangeID field?	8.2.2	M	Yes [ ]
LSlrlaf	—Does the LS support use of the LoadAddress field?	8.2.2	M	Yes [ ]
LSlriad	—Does the LS support use of individual LD addresses in the LoadAddress field?	8.2.2	M	Yes [ ]
LSlrgad	—Does the LS support use of group MAC addresses in the LoadAddress field?	8.2.2	M	Yes [ ]
LSlrdef	—If no LoadAddress is specified in a request, what address will the LS use for the subsequent load? (The LS may use the individual address of the LD or a group MAC address. If group MAC addresses are supported, the address value(s) shall be stated. If the LS supports both, the rule that the LS uses to determine which is used shall be stated.)	8.2.2	M	Individual [ ] Group [ ] (values_____ rule_____ _____)

Item	Feature	References	Status	Support
LSlrbs	—Is the range of values for BlockSize supported by the LS in the range 1–32 767?	8.2.2	M	From__ to__
LSlrmbd	—Is the range of values of MinBlockDelay supported by the LS in the range 1–32 767?	8.2.2	M	From__ to__
LSlrmxd	—Is the range of values of MaxBlockDelay supported by the LS in the range 1–32 767?	8.2.2	M	From__ to__
LSlrlres	—Does the LS support use of the LoadReason field?	8.2.2	M	Yes [ ]
LSlrlinf	—Does the LS support use of the LoadInfo field?	8.2.2	M	Yes [ ]
	In received GroupStatus PDUs:			
LSgsrid	—Does the LS support the use of the ReferenceID field?	8.4.2	M	Yes [ ]
LSgsgn	—Does the LS support the use of the GroupNumber field?	8.4.2	M	Yes [ ]
LSgsrb	—Are both the BitMap and RequiredBlocksCode variants of RequiredBlocks supported?	8.4.2	M	Yes [ ]
	In transmitted LoadResponsePDUs:			
LSlrseid	—Does the LS support the use of the ExchangeID field?	8.2.2, 8.2.3	M	Yes [ ]
LSlrsla	—Does the LS support the use of both individual and group addresses as the destination address for a load?	8.2.2, 8.3.2	M	Yes [ ]
LSlrsbs	—Is the value of BlockSize provided by the LS less than or equal to the value of BlockSize in the corresponding LoadRequestPDU?	8.3.2	M	Yes [ ]
LSlrsmbd	—Is the value of MinBlockDelay provided by the LS greater than or equal to the value of MinBlockDelay in the corresponding LoadRequestPDU?	8.2.2, 8.3.2	M	Yes [ ]
LSlrsmxd	—Is the value of MaxBlockDelay provided by the LS less than or equal to the value of MaxBlockDelay in the corresponding LoadRequestPDU?	8.2.2, 8.3.2	M	Yes [ ]
LSlrsrid	—Does the LS support the use of the ReferenceID field?	8.3.2	M	Yes [ ]
LSlrsnb	—Is the range of values of NumberBlocks supported by the LS 1–65 535?	8.3.2	M	Yes [ ]

Item	Feature	References	Status	Support
LSlrsls	—Is the range of values of LoadSelector supported by the LS in the range –128 to +127?	8.3.2, 8.3.3	O	From__ to__
LSlrsii	—What values does the LS support for the ImageInfo field?	8.3.2	O	Values___
	In transmitted GroupStatusRequestPDUs:			
LSgsrda	—Does the LS support the use of both individual and group MAC addresses as the destination address?	8.5.1	M	Yes [ ]
LSgsrrid	—Does the LS support the use of the ReferenceID field?	8.5.2	M	Yes [ ]
	In transmitted LoadDataPDUs:			
LSlorid	—Does the LS support the use of the ReferenceID field?	8.6.2	M	Yes [ ]
LSlogn	—Is the range of values supported for GroupNumber in the range 0–255?	8.6.2	M	From__ to__
LSlobl	—Is the range of values supported for Block in the range 0–255?	8.6.2	M	From__ to__
LSlold	—Does the LS support the use of the LoadData field?	8.6.2	M	Yes [ ]

A.8 Loadable Device management support

If management of the Loadable Device is not supported, item MOld, mark N/A and continue at A.9. **N/A [ ]**

Item	Feature	References	Status	Support
LDMmo	Does the implementation support the LD and Resource Type ID managed object classes and their attributes and operations?	9.1.5, 9.2.1, 9.2.1.2, 9.3.1, 9.3.3	M	Yes [ ]
LDMstid	Does the LD support the use of the StationID parameter of the Load action PDU?	9.2.1.3, 9.3.1.12	M	Yes [ ]
LDMimid	Does the LD support the use of the ImageID parameter of the Load action PDU?	9.2.1.3, 9.3.1.12	M	Yes [ ]
LDMext	Does the LD support the use of the Extension parameter of the Load action PDU?	9.2.1.3, 9.3.1.12	O	Yes [ ] No [ ]
LDMnam	Does the LD support the LD naming attribute?	9.3.1.1	M	Yes [ ]
LDMcp	Does the LD support the optional attributes identified by the conditional package pLD2?	9.3.1	O	Yes [ ] No [ ]
	SUPPORT OF OPTIONAL ATTRIBUTES If the optional attributes package is not supported, item LDMcp, mark N/A and continue at A.9.			N/A [ ]
LDMli	Does the LD Load Info attribute support the same range of values as is supported for the LoadInfo field in the LoadRequestPDU?	8.2.2, 9.2.1.2, 9.3.1.2	M	Yes [ ]
LDMla	What value(s) of address are supported by the LD Load Server Address attribute?	9.2.1.2, 9.3.1.3	M	Values _____
LDMt1	What value range (in milliseconds) is supported for the LD T1 attribute?	9.2.1.2, 9.3.1.4	M	From ____ to ____
LDMt2	What value range (in milliseconds) is supported for the LD T2 attribute?	9.2.1.2, 9.3.1.5	M	From ____ to ____
LDMrc1	What value range is supported for the LD Retry Count 1 attribute?	9.2.1.2, 9.3.1.6	M	From ____ to ____

Item	Feature	References	Status	Support
LDMrc2	What value range is supported for the LD Retry Count 2 attribute?	9.2.1.2, 9.3.1.7	M	From ____ to ____
LDMbs	Does the LD Block Size attribute support the same range of values as is supported for the BlockSize field in the LoadRequestPDU?	8.2.2, 9.2.1.2, 9.3.1.8	M	Yes [ ]
LDMmbd	Does the LD Min Block Delay attribute support the same range of values as is supported for the MinBlockDelay field in the LoadRequestPDU?	8.2.2, 9.2.1.2, 9.3.1.9	M	Yes [ ]
LDMmxd	Does the LD Max Block Delay attribute support the same range of values as is supported for the MaxBlockDelay field in the LoadRequestPDU?	8.2.2, 9.2.1.2, 9.3.1.10	M	Yes [ ]
LDMsta	Does the LD managed object support the LD Status attribute?	8.2.2, 9.2.1.2, 9.3.1.11	M	Yes[ ]

A.9 Load Server management support

If management of the Load Server is not supported, item MOls, mark N/A and ignore the remainder of A.9.
N/A [ ]

Item	Feature	References	Status	Support
LSMmo	Does the implementation support the LS and Resource Type ID managed object classes and their attributes and operations?	9.1.5, 9.2.2, 9.2.2.2, 9.3.2, 9.3.3	M	Yes [ ]
LSMnam	Does the LD support the LS naming attribute?	9.3.2.1	M	Yes [ ]
LSMt1	What value range (in milliseconds) is supported for the LS T1 attribute?	9.2.2.2, 9.3.2.2	M	From ____ to ____
LSMt2	What value range (in milliseconds) is supported for the LS T2 attribute?	9.2.2.2, 9.3.2.3	M	From ____ to ____
LSMnd	What value range is supported for the LS Net Delay attribute?	9.2.2.2, 9.3.2.4	M	From ____ to ____
LSMrc	What value range is supported for the LS Retry Count attribute?	9.2.2.2, 9.3.2.5	M	From ____ to ____
LSMsta	Does the LS support the LS Status attribute?	9.2.2.2, 9.3.2.6	M	Yes [ ]
LSMstid	Does the LS support the use of the Address parameter of the LSActive event report PDU?	9.2.2.3, 9.3.2.7	M	Yes [ ]
LSMext	Does the LS support the use of the Extension field of the LSActive event report PDU?	9.2.2.3, 9.3.2.7	O	Yes [ ] No [ ]

Annex B
Allocation of object identifier values

(normative)

B.1 Introduction

This annex contains a summary of all object identifier values that have been allocated by this International Standard, both in this revision and in previous revisions.

Each table shows allocations related to a specific category of information object. The heading of the table identifies the category of information object, and shows the invariant part of the object identifier value allocated to the entries in the table. The column labelled Arc shows the value allocated to the arc subsequent to the invariant part, which completes the object identifier value allocated. The column labelled Purpose contains a text description of the information object, and, in the case of current allocations, a reference to the location of the definition of the information object in the standard. The column labelled Status shows the status of the allocated values, using the following convention:

R Reserved. The object identifier value is reserved for future use by this standard.
C Current. The object identifier value has been allocated to an information object that is defined within the current revision of the standard.
D Deprecated. The object identifier value has been allocated to an information object that was defined in a previous revision of the standard, and whose use is now deprecated.

B.2 Allocation tables

Allocations for standard-specific extensions.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) standardSpecificExtensions(0)}

ARC	PURPOSE	STATUS
None allocated	N/A	N/A

Allocations for ASN.1 module identifiers.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) asn1Module(2)}

ARC	PURPOSE	STATUS
loadprotocol(0) version1(0)	ASN.1 module for Load Protocol PDU definitions	C
loaddefinitions(1) version1(0)	ASN.1 module for Attribute, Action and Notification syntaxes	C

Allocations for Managed object classes.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) managedObjectClass(3)}

ARC	PURPOSE	STATUS
ldclass(0)	LD managed object class name	C
lsclass(1)	LS managed object class name	C

Allocations for Package identifiers.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) packages(4)}

ARC	PURPOSE	STATUS
pld2(0)	LD optional attributes package	C

Allocations for Parameters.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) parameter(5)}

ARC	PURPOSE	STATUS
None Allocated	N/A	N/A

Allocations for Name Binding identifiers.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) nameBinding(6)}

ARC	PURPOSE	STATUS
ldbinding(0)	Name binding, LD managed object to system	C
lsbinding(1)	Name binding, LS managed object to system	C
resourcetypeID-lsbinding(2)	Name binding, Resource Type ID to LS	C
resourcetypeID-ldbinding(3)	Name binding, Resource Type ID to LS	C

Allocations for Attribute identifiers.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)}

ARC	PURPOSE	STATUS
ldname(0)	LD naming attribute	C
loadinfo(1)	LD Load Info attribute	C
ldloadserveraddress(2)	LD Load Server Address attribute	C
ldt1(3)	LD T1 attribute	C
ldt2(4)	LD T2 attribute	C

Allocations for Attribute identifiers. (*Continued*)
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) attribute(7)}

ARC	PURPOSE	STATUS
ldretrycount1(5)	LD Retry Count 1 attribute	C
ldretrycount2(6)	LD Retry Count 2 attribute	C
ldblocksize(7)	LD Block Size attribute	C
ldminblockdelay(8)	LD Min Block Delay attribute	C
ldmaxblockdelay(9)	LD Max Block Delay attribute	C
ldstatus(10)	LD Status attribute	C
lsname(11)	LS naming attribute	C
lst1(12)	LS T1 attribute	C
lst2(13)	LS T2 attribute	C
lsnetdelay(14)	LS Net Delay attribute	C
lsretrycount(15)	LS Retry Count attribute	C
lsstatus(16)	LS Status attribute	C

Allocations for Attribute Group identifiers.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) attributeGroup(8)}

ARC	PURPOSE	STATUS
None Allocated	N/A	N/A

Allocations for Action types.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) action(9)}

ARC	PURPOSE	STATUS
load(0)	Load action identifier	C

Allocations for Notification types.
Invariant part of object identifier value =
{iso(1) member-body(2) us(840) ieee802dot1partE(10010) notification(10)}

ARC	PURPOSE	STATUS
lsactive(0)	LS Active notification	C

Annex C
System issues

(informative)

C.1 Load sharing

If more than one LS sends a LoadResponsePDU in answer to a LoadRequestPDU, the LD selects the LS that is to provide the load by sending it a GroupStatusPDU in response. The GroupStatusPDU is sent to the LS individual MAC address, so only one LS receives the GroupStatusPDU and subsequently loads the LD. All other LS's that have responded to the request will timeout waiting for a GroupStatusPDU and will not load the requesting LD.

The LoadSelector field of the LoadResponsePDU may be used in order to select an LS. For example, if more than one LS may load an LD and if one of them is currently performing a load of the requested image, it could increase its LoadSelector based on this knowledge. Alternatively, if the LS is heavily loaded with other operations, but is capable of performing the load, it might lower the selector value to defer if possible.

If an LS or a set of LS's fails, an LD that requests its load from those LS's can automatically obtain a load from one of the other members of the set of LS's by re-requesting the load.

A byproduct of the loadsharing and recovery capabilities is the general support of redundant servers. Redundant LS's might be assigned different LoadSelector values. An LD might have one LS as its primary sender and select that LS based on the LoadSelector. It might select a backup LS if its primary LS was not available.

If all LS's from which an LD can obtain an image fail, the LD will not be able to obtain its load unless one or more of the LD's load-related parameters (address, image, delay, etc.) is modified so as to cause a different set of LS's to respond to the request.