IEEE Standards

IEEE Std 802.1X™- 2004
(Revision of
IEEE Std 802.1X-2001)

802.1X™

IEEE Standard for
Local and metropolitan area networks

Port-Based Network Access Control

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE

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

13 December 2004

Print: SH95298
PDF: SS95298

IEEE Std 802.1X™-2004
(Revision of
IEEE Std 802.1X-2001)

IEEE Standard for
Local and metropolitan area networks

Port-Based Network Access Control

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 15 November 2004

IEEE-SA Standards Board

Abstract: Port-based network access control makes use of the physical access characteristics of IEEE 802® Local Area Networks (LAN) infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases in which the authentication and authorization process fails.
Keywords: authentication, authorization, controlled port, local area networks, metropolitan area networks, port access control, uncontrolled port

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

Print: ISBN 0-7381-4528-8 SH95298
PDF: ISBN 0-7381-4529-7 SS95298

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

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

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

Secretary, IEEE-SA Standards Board

445 Hoes Lane

P.O. Box 1331

Piscataway, NJ 08855-1331USA

NOTE—Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Introduction

[This introduction is not part of IEEE Std 802.1X-2004, IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control.]

This standard defines a mechanism for Port-based network access control that makes use of the physical access characteristics of IEEE 802 LAN infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases in which the authentication and authorization process fails.

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Participants

At the time this standard was revised, the working group had the following membership:

Tony Jeffree, *Chair and Editor*

Paul Congdon, *Vice-Chair*

Dolors Sala, *Chair, Link Security Task Group*

Floyd Backes
Les Bell
Paul Bottorff
Laura Bridge
Jim Burns
Dirceu Cavendish
Hesham Elbakoury
Norm Finn
David Frattura
Gerard Goubert
Steve Haddock
Ran Ish-Shalom
Neil Jarvis
Hal Keen
Bill Lane
Roger Lapuh
Loren Larsen
Dinesh Mohan
Bob Moskowitz
Don O'Connor
Glenn Parsons
Frank Reichstein
John Roese
Allyn Romanow
Dan Romascanu
Mick Seaman
Kazuo Takagi
Michel Thorsen
Dennis Volpano
Karl Weber
Michael D. Wright

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

Butch Anton
Reza Arefi
William Byrd
John Barnett
Hugh Barrass
Barbara Bickham
Gennaro Boggia
Todd Cooper
Edward Carley Jr.
Clifford Chamney
Kai Moon Chow
Keith Chow
Todor Cooklev
Javier del Prado Pavon
Guru Dutt Dhingra
Wael Diab
Thomas Dineen
Lakshminath Dondeti
Dr. Sourav Dutta
Clint Early, Jr.
Jonathan Edney
Darwin Engwer
Matthew Fischer
Will Foulds
Ernesto Garcia
Jean-Denis Gorin
Ron Greenthaler
Luigi Grieco
Robert Grow
Chris Guy
David Halasz
Matt Holdrege
Stuart Holoman
David Hunter
Atsushi Ito
Peeya Iwagoshi
Raj Jain
David James
Neil Jarvis
Tony Jeffree
David Johnston
Bobby Jose
Joe Juisai
Thomas M. Kurihara
Joerg Kampmann
Junghong Kao
Kevin Karcz
Piotr Karocki
Ken Katano
Byoung-Jo Kim
Cees Klik
Pi-Cheng Law
Juanluis Lazaro
Armando Leite
Panos Lekkas
John Lemon
Randolph Little
Henry Lu
William Lumpkins
Nikolai Malykh
Ben Manny
Jacques Mathot
Kyle Maus
Jonathon McLendon
Jorge Medina
Steve Methley
David Mitton
Apurva Mody
Leo Monteban
Jeremy Moore
Joseph Moran
Mike Moreton
Robert Moskowitz
Narayanan Murugesan
Andrew Myles
Charles Ngethe
Nick S.A. Nikjoo
Donald O'Connor
Bob O'Hara
Satoshi Obara
Chris Osterloh
Stephen Palm
Leo Pluswick
Subbu Ponnuswamy
Vikram Punj
maximilian Riegel
John Roese
Floyd Ross
Vaughn Sheline
Gil Shultz
Amjad Soomro
Adrian Stephens
Masahiro Takagi
Joseph Tardo
Mark-Rene Uchida
Scott Valcourt
Tom Wandeloski
Stanley Wang
William Ward
Dave Willow
Derek Woo
Eric Woods
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 15 November 2004, it had the following membership:

Don Wright, *Chair*
Steve M. Mills, *Vice Chair*
Judith Gorman, *Secretary*

Chuck Adams
H. Stephen Berger
Mark D. Bowman
Joseph A. Bruder
Bob Davis
Roberto de Boisson
Julian Forster*
Arnold M. Greenspan
Mark S. Halpin
Raymond Hapeman
Richard J. Holleman
Richard H. Hulett
Lowell G. Johnson
Joseph L. Koepfinger*
Hermann Koch
Thomas J. McGean
Daleep C. Mohla
Paul Nikolich
T. W. Olsen
Ronald C. Petersen
Gary S. Robinson
Frank Stone
Malcolm V. Thaden
Doug Topping
Joe D. Watson

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Alan Cookson, *NIST Representative*

Michelle D. Turner
IEEE Standards Project Editor

Historical participants

Since the initial publication, many IEEE standards have added functionality or provided updates to material included in this standard. The following is a historical list of participants who have dedicated their valuable time, energy, and knowledge to the creation of this material:

Les Bell
Alan Chambers
Marc Cochran
Paul Congdon
Hesham El Bakoury
Norman W. Finn
Sharam Hakimi
Bob Hott
Neil Jarvis
Tony Jeffree
Toyoyuki Kato
Hal Keen
Daniel Kelley
Keith Klamm
Joe Laurence
Bill Lidinsky
Yaron Nachman
LeRoy Nash
Satoshi Obara
Luc Pariseau
Anil Rijsinghani
John J. Roese
Ted Schroeder
Benjamin Schultz
Mick Seaman
Rosemary V. Slager
Andrew Smith
Michel Soerensen
Robin Tasker
Manoj Wadekar
Robert Williams

Contents

IEEE Standard for
Local and metropolitan area networks

Port-Based Network Access Control

1. Overview

1.1 Scope

IEEE 802® Local Area Networks (or LANs; see 3.4 in IEEE Std 802.1D™) are often deployed in environments that permit unauthorized devices to be physically attached to the LAN infrastructure, or permit unauthorized users to attempt to access the LAN through equipment already attached. Examples of such environments include corporate LANs that provide LAN connectivity in areas of a building that are accessible to the general public, and LANs that are deployed by one organization in order to offer connectivity services to other organizations (for example, as may occur in a business park or a serviced office building). In such environments, it is desirable to restrict access to the services offered by the LAN to those users and devices that are permitted to make use of those services.

Port-based network access control makes use of the physical access characteristics of IEEE 802 LAN infrastructures in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases in which the authentication and authorization process fails. A port in this context is a single point of attachment to the LAN infrastructure. Examples of ports in which the use of authentication can be desirable include the Ports of Media Access Control (MAC) Bridges (as specified in IEEE Std 802.1D), the ports used to attach servers or routers to the LAN infrastructure, and associations between stations and access points in IEEE 802.11™ Wireless LANs.

1.2 Purpose

For the purpose of providing compatible authentication and authorization mechanisms for devices interconnected by IEEE 802 LANs, this standard specifies a general method for the provision of port-based network access control. To this end, it

a) Describes the architectural framework within which the authentication, and consequent actions, take place

b) Defines the principles of operation of the access control mechanisms

c) Defines the different levels of access control that are supported, and the behavior of the port with respect to the transmission and reception of frames at each level of access control

d) Establishes the requirements for a protocol between the device that requires the authentication to take place (the Authenticator; see 3.1.1) and the device that is attached to the Authenticator's port (the Supplicant; see 3.1.12)

e) Establishes the requirements for a protocol between the Authenticator and an Authentication Server (see 3.1.4)

f) Specifies mechanisms and procedures that support network access control through the use of authentication and authorization protocols

g) Specifies the encoding of the Protocol Data Units (PDUs) used in authentication and authorization protocol exchanges

h) Establishes the requirements for management of port-based access control, identifying the managed objects and defining the management operations

i) Specifies how the management operations are made available to a remote manager using the protocol and architectural description provided by the Simple Network Management Protocol (SNMP) (IETF RFC 3411)

j) Specifies the requirements to be satisfied by equipment claiming conformance to this standard

2. References

The following standards contain provisions which, through reference in this text, constitute provisions of the standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of ISO and IEC maintain registers of currently valid International Standards.

ANSI X3.159-1989, American National Standards for Information Systems—Programming Language—C.[1]

IEEE Std 802.1D™-2004, IEEE Standard for Local and metropolitan area networks: Media access control (MAC) Bridges.[2, 3]

IEEE Std 802.1Q™, 2003 Edition, IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks.

IEEE Std 802.3™-2002, IEEE Standard for Information technology—Local and metropolitan area networks—Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

IEEE Std 802.5™, 1998 Edition (ISO/IEC 8802-5-1998), IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 5: Token ring access method and physical layer specifications. [4]

IEEE Std 802.11™, 2003 Edition (ISO/IEC 8802-11: 2003), IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

IETF RFC 1305, Network Time Protocol (Version 3) Specification, Implementation and Analysis, Mills, D. L., March 1992.[5]

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

IETF RFC 2104, HMAC: Keyed-Hashing for Message Authentication, Krawczyk, H., Bellare, M., and Canetti, R., February 1997.

IETF RFC 2433, Microsoft PPP CHAP Extensions, G. Zorn, S. Cobb, October 1998.

IETF RFC 2548, Microsoft Vendor-specific RADIUS Attributes, G. Zorn, March 1999

---

[1]ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[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. IEEE publications can be ordered on-line from the IEEE Standards Website: http://www.standards.ieee.org.

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

[4]IEEE [ISO] and IEEE [ISO/IEC] documents are available from ISO Central Secretariat, 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse; and from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. IEEE [ISO] and IEEE [ISO/IEC] documents can be ordered on-line from the IEEE Standards Website: http://www.standards.ieee.org.

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

IETF RFC 2578, STD 58, Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., Waldbusser, S., April 1999.

IETF RFC 2579, STD 58, Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2), McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M., Waldbusser, S., April 1999.

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

IETF RFC 2607, Proxy Chaining and Policy Implementation in Roaming, B. Aboba, J. Volbrecht, June 1999.

IETF RFC 2716, PPP EAP TLS Authentication Protocol, Aboba, B. and Simon, D., October 1999.

IETF RFC 2863, The Interfaces Group MIB using SMIv2, McCloghrie, K. and Kastenholz, F., June 2000.

IETF RFC 2865, Remote Authentication Dial In User Service (RADIUS), Rigney, C., Willens, S., Rubens, A., and Simpson, W., June 2000.

IETF RFC 2866, RADIUS accounting, Rigney, C., June 2000.

IETF RFC 2867, RADIUS Accounting Modifications for Tunnel Protocol Support, Zorn, G., Aboba, B., and Mitton, D., June 2000.

IETF RFC 2868, RADIUS Attributes for Tunnel Protocol Support, Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and Goyret, I., June 2000.

IETF RFC 2869, RADIUS Extensions, Rigney, C., Willats, W., and Calhoun, P., June 2000.

IETF RFC 3162, RADIUS and IPv6, B. Aboba, G. Zorn, D. Mitton, August 2001.

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

IETF RFC 3411, An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, D. Harrington, R. Presuhn, B. Wijnen, December 2002.

IETF RFC 3412, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), J. Case, D. Harrington, R. Presuhn, B. Wijnen, December 2002.

IETF RFC 3413, Simple Network Management Protocol (SNMP) Applications, D. Levi, P. Meyer, B. Stewart, December 2002.

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

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

IETF RFC 3416, Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP), R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, December 2002.

IETF RFC 3417, Transport Mappings for the Simple Network Management Protocol (SNMP), R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, December 2002.

IETF RFC 3418, Management Information Base (MIB) for the Simple Network Management Protocol (SNMP), R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, December 2002.

IETF RFC 3575, IANA Considerations for RADIUS, B. Aboba, July 2003.

IETF RFC 3576, Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS), M. Chiba, G. Dommety, M. Eklund, D. Mitton and B. Aboba, July 2003.

IETF RFC 3579, RADIUS Support for Extensible Authentication Protocol (EAP), B. Aboba, P. Calhoun, September 2003.

IETF RFC 3580, IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines, P.Congdon, September 2003.

IETF RFC 3748, Extensible Authentication Protocol (EAP), Blunk, L., Vollbrecht, J., Aboba, B., Carlson, J., Levkowetz, H, June 2004.

ISO/IEC 8824:1990, Information technology—Open Systems Interconnection—Specification of Abstract Syntax Notation One (ASN.1) (Provisionally retained edition).[6]

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

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

3. Definitions

For the purposes of this standard, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms,* Seventh Edition [B5][7], should be referenced for terms not defined in this clause.

3.1 Terms defined in this standard

3.1.1 Authenticator: An entity at one end of a point-to-point LAN segment that facilitates authentication of the entity attached to the other end of that link.

3.1.2 authentication exchange: The two-party conversation between systems performing an authentication process.

NOTE—For example, Extensible Authentication Protocol (EAP) and Simple Authentication and Security Layer (SASL).[8]

3.1.3 authentication process: The cryptographic operations and supporting data frames that perform the actual authentication.

3.1.4 Authentication Server: An entity that provides an authentication service to an Authenticator. This service determines, from the credentials provided by the Supplicant, whether the Supplicant is authorized to access the services provided by the system in which the Authenticator resides.

NOTE—The Authentication Server function can be co-located with an Authenticator, or it can be accessed remotely via a network to which the Authenticator has access.

3.1.5 authentication transport: The datagram session that actively moves the authentication exchange between two systems.

NOTE—Examples include EAP Over LANs (EAPOL), Remote Authentication Dial In User Service (RADIUS), and Diameter (see IETF RFC 3588 [B4]).

3.1.6 Bridge Port: A Port of an IEEE 802.1D or IEEE 802.1Q Bridge.

3.1.7 Edge Port: A Bridge Port attached to a LAN that has no other Bridges attached to it.

NOTE—See 17.3 in IEEE Std 802.1D.

3.1.8 frame: MAC protocol data unit (MPDU).

3.1.9 message digest: The output produced by applying a hash function to a message.

3.1.10 network access port: A point of attachment of a system to a LAN. It can be a physical port, for example, a single LAN MAC attached to a physical LAN segment, or a logical port, for example, an IEEE 802.11 association between a station and an access point.

NOTE—The term *Port* is used in this standard as an abbreviation of network access port (see 3.3.2).

3.1.11 port access entity (PAE): The protocol entity associated with a Port. It can support the protocol functionality associated with the Authenticator, the Supplicant, or both.

[7]The numbers in brackets correspond to those of the bibliography in Annex F.

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

3.1.12 Supplicant: An entity at one end of a point-to-point LAN segment that seeks to be authenticated by an Authenticator attached to the other end of that link.

NOTE—The term *Supplicant* is used in this standard in place of the more conventional term, *peer*, used in other access control-related specifications.

3.1.13 system: A device that is attached to a LAN by one or more Ports. Examples of systems include end stations, servers, MAC Bridges, and routers.

3.2 Terms defined in IEEE Std 802.1D

3.2.1 Bridged Local Area Network: A concatenation of individual IEEE 802 LANs interconnected by MAC Bridges.

3.2.2 IEEE 802 Local Area Network (LAN): IEEE 802 LANs (also referred to in the text simply as LANs) are LAN technologies that provide a MAC Service equivalent to the MAC Service defined in ISO/IEC 15802-1. IEEE 802 LANs include IEEE Std 802.3 (CSMA/CD), IEEE Std 802.5 (Token Ring), IEEE Std 802.11 (Wireless), and ISO 9314-2 (FDDI) LANs.

3.3 Terms defined in *The Authoritative Dictionary of IEEE Standards Terms* [B5]

3.3.1 nonce

3.3.2 Port

4. Acronyms and abbreviations

For the purposes of this standard, the following acronyms and abbreviations apply. *The Authoritative Dictionary of IEEE Standards Terms* [B5], should be referenced for terms not defined in this clause.

AAA	authentication, authorization, and accounting
ASF	Alerting Standards Forum
CHAP	Challenge Handshake Authentication Protocol
DHCP	Dynamic Host Configuration Protocol
EAP	extensible authentication protocol
EAP-TLS	EAP Transport Layer Security
EAPOL	EAP over LANs
FDDI	Fiber Distributed Data Interface
IP	Internet Protocol
LAN	IEEE 802 Local Area Network
LLC	Logical Link Control
MAC	media access control
MPDU	MAC protocol data unit
MSCHAP	Microsoft Challenge Handshake Authentication Protocol
PACP	Port Access Control Protocol
PAE	port access entity
PID	Protocol Identifier
Port	network access port
PDU	protocol data unit
RADIUS	remote authentication dial in user service
RIF	Routing Information Field
SASL	Simple Authentication and Security Layer
SNAP	Subnetwork Access Protocol
SNMP	Simple Network Management Protocol
VLAN	Virtual LAN

5. Conformance

5.1 Requirements

A device for which conformance to this standard is claimed shall, for all Ports for which support is claimed:

a) Support the operation of the Port Access Entity (PAE) over the uncontrolled Port, as a Supplicant PAE, an Authenticator PAE, or both, as defined in Clause 8

b) Support the system configuration functions as defined in 9.6.1

c) Support operation of the controlled Port in a manner consistent with the use of AuthControlledPortControl parameter values of Force Unauthorized, Auto and Force Authorized, as defined in 6.4

d) Support the ability to set the AuthControlledPortControl parameter to the values of Force Unauthorized, Auto and Force Authorized, as defined in 6.4, by management action

e) Support operation of the controlled Port in a manner consistent with the use of AdminControlledDirections and OperControlledDirections parameter values of Both, as defined in 6.5

f) Where Authenticator PAE operation is supported:

   1) Support the ability to configure the operation of the Authenticator as defined in 9.4.1

   2) Support the ability to maintain and retrieve the Authenticator statistics as described in 9.4.2

   3) Support regular reauthentication of the Supplicant by means of the Reauthentication Timer state machine, and support the ability to modify the reAuthTimer and reAuthEnabled parameters by management action (see 8.2.8 and 9.4.1)

g) Where Supplicant PAE operation is supported:

   1) Support the ability to configure the operation of the Supplicant as defined in 9.5.1

   2) Support the ability to maintain and retrieve the Supplicant statistics as described in 9.5.2

h) Where both Authenticator PAE operation and Supplicant PAE operation is supported, support a *Supplicant Access Control With Authenticator* administrative control parameter value of inactive (see 6.4, 8.2.2.2)

5.2 Options

A device for which conformance to this standard is claimed may, for any Port for which support is claimed:

a) Support the operation of protocol entities other than the PAE over the uncontrolled Port

b) Support operation of the controlled Port in a manner consistent with the use of AdminControlledDirections and OperControlledDirections parameter values of In, and support the ability to set the AdminControlledDirections parameter to the values In and Both by management action, as defined in 6.5

c) Where Authenticator PAE operation is supported:

   1) Support the ability to maintain and retrieve the Authenticator diagnostics as described in 9.4.3

   2) Support the ability to maintain and retrieve the Authenticator session statistics as described in 9.4.4

   3) Support the ability to transmit key information to the Supplicant and support the ability to modify the KeyTransmissionEnabled parameter by management action (see 8.1.9, 8.2.5, and 9.4.1)

d) Where Supplicant PAE operation is supported:

   1) Support the ability to transmit key information to the Authenticator, and support the ability to modify the KeyTransmissionEnabled parameter by management action (see 8.1.9, 8.2.6, and 9.4.1)

e) Where both Authenticator PAE operation and Supplicant PAE operation is supported, support *Supplicant Access Control With Authenticator* administrative control parameter values of active and inactive (see 6.4, 8.2.2.2), and the ability to modify the parameter value by management action (see 9.5.1)

6. Principles of Port Access Control operation

This clause describes the architectural framework of Port-based access control and the relationship between the access control function and the operation of the device(s) within which it is deployed.

6.1 Purpose of Port Access Control operation

Port Access Control provides an optional extension to the functionality of a System (see 6.3) that offers a means of preventing unauthorized access by Supplicants (see 3.1.12) to the services offered by that System, and also preventing a Supplicant from attempting to access an unauthorized System. Port Access Control also provides a means whereby a Supplicant system may prevent an unauthorized system from connecting to it. For example, if the System concerned is a MAC Bridge, control over access to the Bridge and the LAN to which it is connected can be desirable in order to restrict access to publicly accessible Bridge Ports, or within an organization, to restrict access to a departmental LAN to members of that department. A Supplicant system can also make use of the outcome of the exchange to prevent unauthorized access.

Access control is achieved by the System enforcing authentication of Supplicants that attach to the System's controlled Ports (see 6.4); from the result of the authentication process, the System can determine whether or not the Supplicant is authorized to access its services on that controlled Port. If the Supplicant is not authorized for access, then both the Supplicant's System and the Authenticator's System set their controlled Port state to unauthorized. In the unauthorized state, the use of the controlled Port is restricted in accordance with the value of the OperControlledDirections parameter associated with that controlled Port (see 6.5), preventing unauthorized data transfer between the Supplicant System and the services offered by the Authenticator System.

The mechanisms defined can be applied to allow any System to authenticate another System that is connected to one of its controlled Ports. The Systems concerned include end stations, servers, routers, and MAC Bridges.

NOTE—Given that the Supplicant device may be a Bridge or a router that wished to connect to and thereby extend a Bridged Local Area Network, the process of authentication will determine whether it can be trusted to connect to the Bridged Local Area Network, given that by doing so, it will effectively be extending the authentication boundary to any other Ports of the Supplicant. Similarly, it may be that, in this situation, there is a need for each device to authenticate the other before either Port can become authorized; see the example shown in Figure 6-6.

6.2 Scope of Port Access Control operation

The operation of Port Access Control assumes that the Ports on which it operates offer a point-to-point connection between a single Supplicant (see 3.1.12) and a single Authenticator (see 3.1.1). It is this assumption that allows the authentication decisions to be made on a per-Port basis. The authentication of multiple Supplicant PAEs attached to a single Authenticator PAE is outside of the scope of this standard.

NOTE 1—For example, the operation of Port Access Control on a classic Token Ring LAN, or on a shared media Ethernet LAN segment, is outside of the scope of this standard.

NOTE 2—IEEE 802.11 LANs can make use of the mechanisms defined in this standard, as such, LANs create a logical port per association, and the point-to-point connection can be enforced by means of suitable encryption.

This standard provides a protocol for communicating authentication information between a Supplicant that is attached to a Port of an Authenticator System and an Authenticator Server, and for controlling the state of the Authenticator and Supplicant System Ports, depending on the outcome of the protocol exchange. This standard does not specify the nature of the authentication information that is exchanged, nor the basis upon which the Authentication Server makes its authentication decisions.

6.3 Systems, Ports, and system roles

Devices that attach to a LAN, referred to in this standard as Systems (see 3.1.13), have one or more points of attachment to the LAN, referred to in this standard as network access ports, or Ports (see 3.1.10).

NOTE 1—An end station generally has a single point of attachment in the form of a network interface card (although some end stations, such as servers, often have multiple points of attachment); a MAC Bridge generally has two or more points of attachment in the form of the Bridge Ports.

The Port(s) of a System provide the means in which it can access services offered by other Systems reachable via the LAN, and provide the means in which it can offer services to, or access the services provided by, other Systems reachable via the LAN. Port-based network access control allows the operation of a System's Port(s) to be controlled in order to ensure that access to its services, and/or access to the services of other Systems, is only permitted by Systems that are authorized to do so.

NOTE 2—The services that a System can offer include the relay function of a MAC Bridge, the routing function of a network layer router, file server functionality, and so on.

For the purposes of describing the operation of Port-based access control, a Port of a System (or more correctly, a PAE associated with a Port; see 6.6) is able to adopt either or both of two distinct roles within an access control interaction:

a) **Authenticator (see 3.1.1):** The Port that wishes to enforce authentication before allowing access to services that are accessible via that Port adopts the Authenticator role.

b) **Supplicant (see 3.1.12):** The Port that wishes to access the services offered by the Authenticator's system adopts the Supplicant role.

A further System role is described as follows:

c) **Authentication server (see 3.1.4):** The Authentication Server performs the authentication function necessary to check the credentials of the Supplicant on behalf of the Authenticator and indicates whether the Supplicant is authorized to access the Authenticator's services.

As can be seen from these descriptions, all three roles are necessary to complete an authentication exchange. A given System can be capable of adopting one or more of these roles; for example, an Authenticator and an Authentication Server can be co-located within the same System, allowing that System to perform the authentication function without the need for communication with an external server. Similarly, a PAE can adopt the Supplicant role in some authentication exchanges, and the Authenticator role in others. An example of the latter may be found in a Bridged Local Area Network, where a new Bridge added to the Bridged Local Area Network may need to be successfully authenticated by the PAE associated with the Port of the Bridge via which it connects to the Bridged Local Area Network before it can authenticate other systems that attach to its Ports. This type of situation is discussed further in 6.7.

NOTE 3—Although co-location of the Authentication Server with an Authenticator is possible, the most common implementation of this mechanism will likely involve the use of an Authentication Server that is external to the Systems that contain the Authenticators.

6.4 Controlled and uncontrolled access

Figure 6-1 illustrates that the operation of Port Access Control has the effect of creating two distinct points of access to the System's point of attachment to the LAN. One point of access allows the uncontrolled exchange of PDUs between the System and other Systems on the LAN, regardless of the authorization state (the uncontrolled Port); the other point of access allows the exchange of PDUs only if the current state of the Port is Authorized (the controlled Port). The uncontrolled and controlled Ports are considered to be part of

the same point of attachment to the LAN; any frame received on the physical Port is made available at both the controlled and uncontrolled Ports, subject to the authorization state associated with the controlled Port.

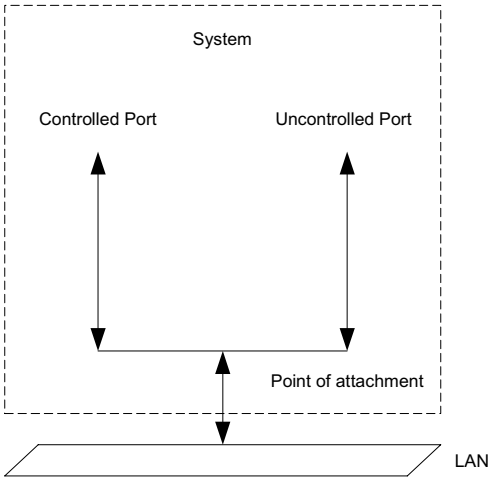


Figure 6-1—Uncontrolled and controlled Ports

NOTE 1—Although a given received frame is made available at both the controlled and uncontrolled Ports, a protocol entity can only be attached to one of the ports at a given moment in time; hence, dependent upon the authorization state of the controlled Port and which Port (if any) the relevant protocol entity is attached to, the frame can be discarded at one or both of the Ports, and is processed by (at most) one protocol entity attached to one of the Ports.

NOTE 2—Arrows are used in this and subsequent diagrams to indicate the connectivity that is available in the various configurations illustrated. For example, in Figure 6-1, the upward pointing arrows indicate that incoming frames can reach users attached to both the controlled and uncontrolled Ports; the downward pointing arrows indicate that outbound frames from either the controlled or uncontrolled Port can reach the LAN.

The point of attachment to the LAN can be provided by any physical or logical Port that can provide a one-to-one connection to another System. For example, the point of attachment could be provided by a single LAN MAC in a switched LAN infrastructure. In LAN environments where the MAC method allows the possibility of a one-to-many relationship between an Authenticator and a Supplicant (for example, in shared media environments), the creation of a distinct association between two Systems is a necessary precondition for the access control mechanisms described in this standard to function. An example of such an association would be an IEEE 802.11 association between a station and an access point.

Figure 6-2 illustrates the effect of the *AuthControlledPortStatus* associated with the controlled Port, representing that status as a switch that can be turned on or off, thus allowing or preventing the flow of PDUs via that Port. The figure shows two systems, each with a single Port; the OperControlledDirections parameter (see 6.5) for each Port is assumed to be set to Both. In System 1, the AuthControlledPortStatus associated with the controlled Port is *unauthorized* and is therefore disabled (the "switch" is turned off); in System 2, the AuthControlledPortStatus is *authorized* and is therefore enabled (the "switch" is turned on).

In addition to the AuthControlledPortStatus, an *AuthControlledPortControl* parameter associated with the Port allows administrative control over the Port's authorization status. This parameter can take the values *ForceUnauthorized, Auto,* and *ForceAuthorized*; its default value is Auto. The relationship between the AuthControlledPortStatus and AuthControlledPortControl parameters is as follows:

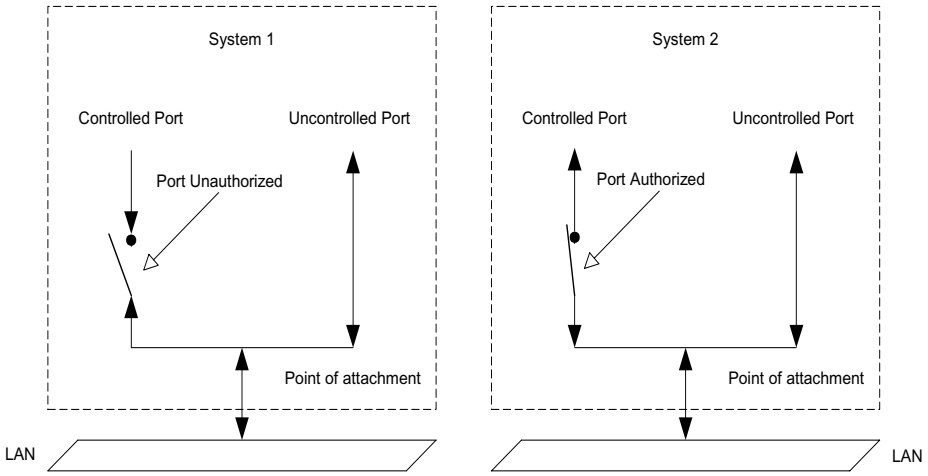


Figure 6-2—Effect of authorization state on controlled Ports

a) An AuthControlledPortControl value of ForceUnauthorized forces the Authenticator PAE state machine (see 8.2.4) to set the value of AuthControlledPortStatus to be unauthorized; i.e., the Controlled Port is unauthorized unconditionally.

b) An AuthControlledPortControl value of ForceAuthorized forces the Authenticator PAE state machine (see 8.2.4) to set the value of AuthControlledPortStatus to be Authorized; i.e., the Controlled Port is authorized unconditionally.

c) An AuthControlledPortControl value of Auto allows the Authenticator PAE state machine (see 8.2.4) to control the value of AuthControlledPortStatus to reflect the outcome of the authentication exchanges between Supplicant PAE, Authenticator PAE, and Authentication Server.

In all three cases, the value of AuthControlledPortStatus directly reflects the value of the portStatus variable maintained by the Authenticator and Supplicant PAE state machines (see 8.2.2.2, 8.2.4, and 8.2.11). Three factors contribute to the value of the portStatus variable:

d) The authorization state of the Authenticator PAE state machine (assumed to be “Authorized” if the state machine is not implemented for that Port).

e) The authorization state of the Supplicant PAE state machine (assumed to be “Authorized” if the state machine is not implemented for that Port).

f) The state of the *Supplicant Access Control With Authenticator* administrative control parameter. This parameter has two possible values, *active* and *inactive*. The default value of this control parameter is *inactive*; support of the *active* value is optional. The value of this parameter takes effect only if both Authenticator PAE and Supplicant PAE state machines are implemented for that Port. If the value of the parameter is *inactive*, then the portStatus parameter value is determined only by the authorization state of the Authenticator PAE state machine. If the value of the parameter is *active*, then the portStatus parameter value is determined by the authorization state of both the Authenticator PAE and Supplicant PAE state machines; if either state machine is in an unauthorized state, then the value of portStatus is unauthorized.

The value of the AuthControlledPortControl parameter for every Port of a System can be overridden by means of the *SystemAuthControl* parameter for the System. This parameter can take the values *Enabled* and *Disabled*; its default value is Disabled. If SystemAuthControl is set to Enabled, then authentication is enabled for the System, and each Port’s authorization status is controlled in accordance with the value of the Port’s AuthControlledPortControl parameter. If SystemAuthControl is set to Disabled, then all Ports behave as if their AuthControlledPortControl parameter is set to ForceAuthorized. In effect, setting the SystemAuthControl parameter to Disabled causes authentication to be disabled on all Ports, and it forces all controlled Ports to be Authorized.

Any access to the LAN is subject to the current administrative and operational state of the MAC (or logical MAC) associated with the Port, in addition to AuthControlledPortStatus. If the MAC is physically or administratively inoperable, then no protocol exchanges of any kind can take place using that MAC on either the controlled or the uncontrolled Port. This is illustrated in Figure 6-3; in system 1, both the controlled and uncontrolled Ports are able to access the LAN, as the controlled Port is authorized, and the MAC providing the point of attachment to the LAN is operable. In system 2, neither the controlled nor the uncontrolled Port can access the LAN, as the MAC providing the point of attachment to the LAN is inoperable. The inoperable state of the MAC has also caused the Authenticator PAE to transition the controlled Port to the Unauthorized state, as shown in Figure 6-3.

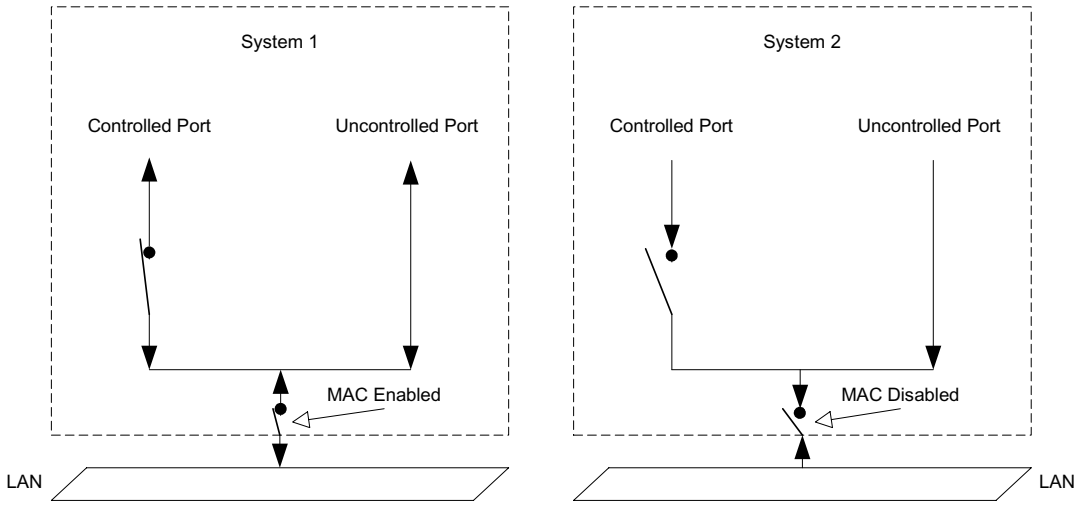


Figure 6-3—Effect of MAC enable/disable states

NOTE 3—Clause 6 of IEEE Std 802.1D describes the parameters available in a Bridge Port that indicate the administrative and operational states associated with the Port's MAC. IEEE Std 802.3 describes similar parameters that define the administrative and operational states associated with the logical MAC offered by an aggregation. The effect of the authorization state on the operability of the Controlled Port is analogous to the effect of the MAC's operational state; however, the effect of the authorization state is modified by the OperControlledDirections parameter associated with the Port, as illustrated in Figure 6-4.

The Authenticator and Supplicant PAEs use the uncontrolled Port for the purposes of exchanging protocol information with another Supplicant or Authenticator PAE.

Protocol exchanges between the Authenticator PAE and the Authentication Server (if the server is not colocated with the Authenticator PAE) can be conducted via one or more of the System's controlled or uncontrolled Ports.

NOTE 4—The details of communication between the Authenticator and the Authentication Server are outside of the scope of this standard. However, such communication would typically be achieved by means of an authentication protocol carried over appropriate higher layer protocols; for example, by means of EAP in RADIUS. Hence, the Authentication Server can be located outside of the confines of the LAN that supports the protocol exchanges between Supplicant and Authenticator, and the communication between the Authenticator and Authentication Server need not be subject to the authentication state of the controlled Port(s) of the systems concerned. If a controlled Port is used to achieve communication with the Authentication Server, protocol exchanges can only take place if the controlled Port is in the authorized state.

It is expected that most protocol exchanges conducted by other functions of the System will make use of one or more of the System's controlled Ports. However, a given protocol may need to bypass the authorization function and make use of the uncontrolled Port. Figure 6-4 shows the uses of the controlled and uncontrolled Ports in an Authenticator System and a Supplicant System, and the ability of the PAEs to change the authorization state of the controlled Port depending on the outcome of an authentication exchange; the figure also gives an example of protocol entities (the PAEs) that require the use of the uncontrolled port in order to conduct their protocol exchanges.

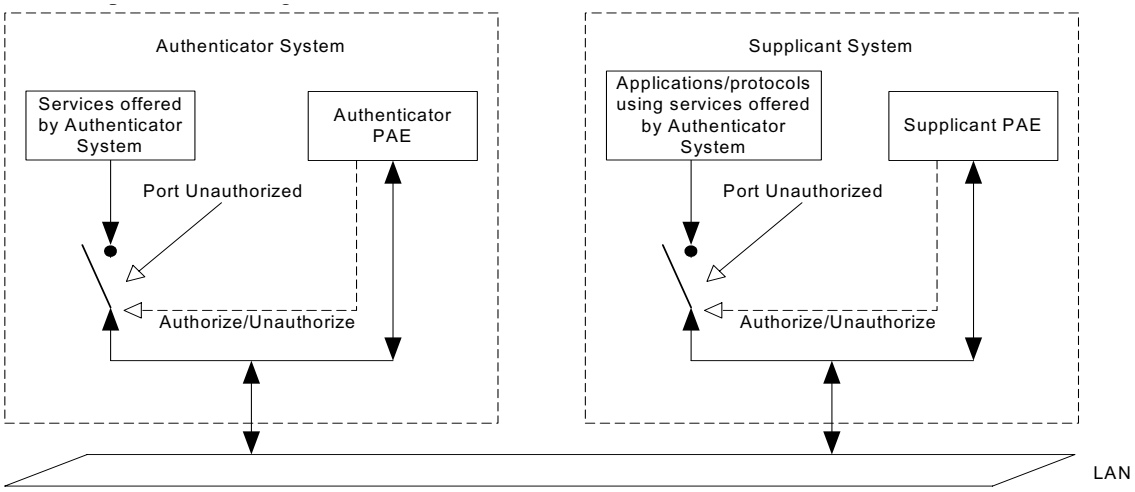


Figure 6-4—Use of the uncontrolled and controlled Ports

Figure 6-5 illustrates the relationships among the Supplicant, Authenticator, and Authentication Server, and the exchange of information among them. In this illustration, both the Authenticator's and the Supplicant's controlled Ports are in the unauthorized state and are therefore disabled from the point of view of access by the Supplicant's System to the services offered by the Authenticator's System. The two PAEs make use of their uncontrolled Ports to communicate with each other, using an authentication protocol carried at the Link Layer, and the Authenticator PAE communicates with the Authentication Server using authentication protocol carried in a higher layer protocol.

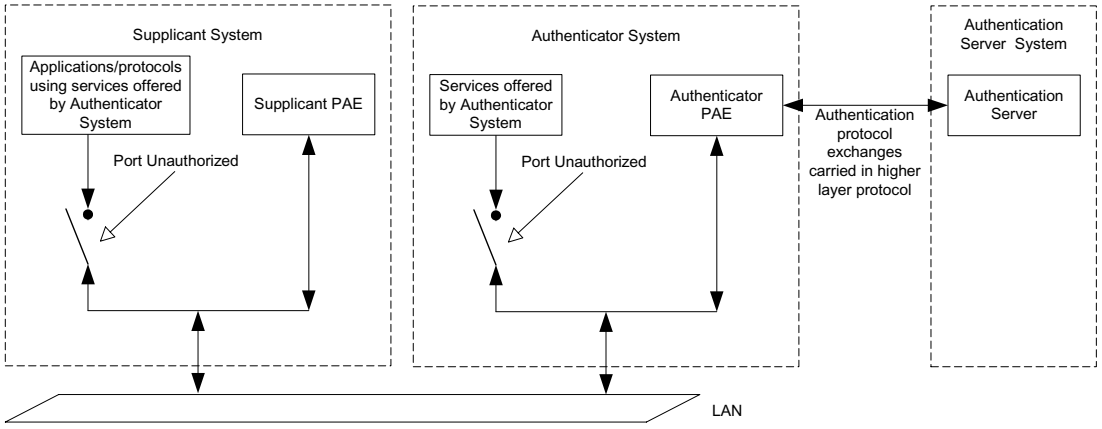


Figure 6-5—Authenticator, Supplicant, and Authentication Server roles

NOTE 5—The situation shown in Figure 6-5 could be found, for example, where the Supplicant's System was an end station and the Authenticator's System was a Bridge at the edge of a Bridged Local Area Network.

The communication between the Authenticator and the Authentication Server may make use of the services of a LAN, or it may use some other communication channel. In cases in which the Authentication Server is co-located with the Authenticator, authentication protocol exchanges between these two entities are unnecessary.

Figure 6-6 illustrates a situation in which the PAEs associated with the two systems, A and B, are able to adopt either the Supplicant or the Authenticator roles, as necessary. In order for System A to make use of

System B's services, System A's PAE must adopt the Supplicant role, and System B's PAE the Authenticator role. For System B to make use of System A's services, the roles are reversed. Note that although the Authentication Server function is shown as residing in two distinct systems in this example, this need not be the case. Note also that, since System A and System B implement both Authenticator and Supplicant PAEs, for the System's controlled Port to become Authorized, the Supplicant and the Authenticator state machines associated with that Port both have to be in the Authorized state.

NOTE 6—This can be visualized by considering the "switch" that represents the authorization state of the Port to actually consist of two individual switches in series, one controlled by the state of the Authenticator PAE for that Port, the other controlled by the state of the Supplicant PAE for that Port.

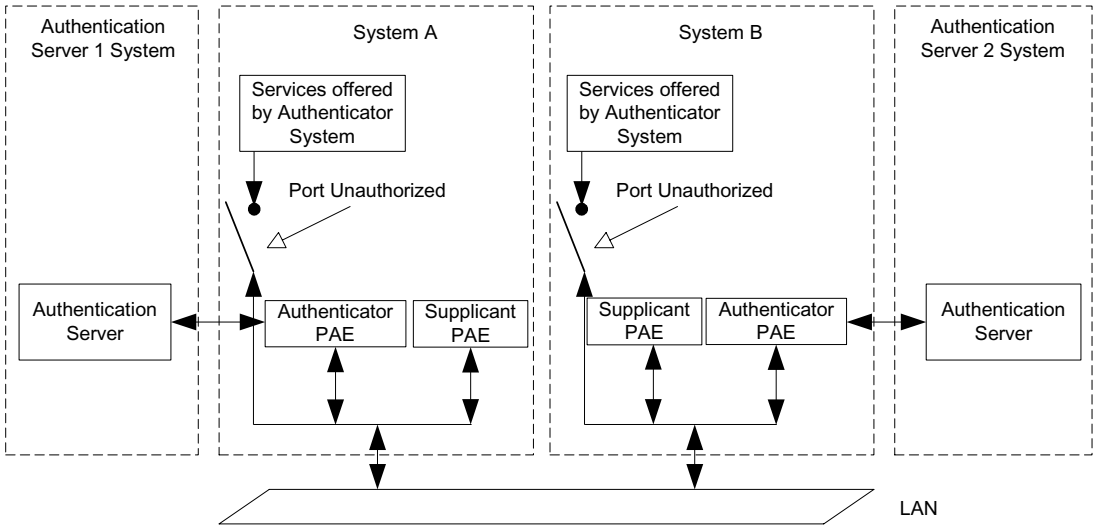


Figure 6-6—Systems adopting both Authenticator and Supplicant roles

NOTE 7—The situation shown in Figure 6-6 could be found, for example, where System A and System B are both Bridges. When they are initially connected together, each Bridge requires the other Bridge to be authenticated and authorized before it will forward frames on behalf of the other Bridge.

In general, the configuration shown in Figure 6-6 is intended for use where there is no obvious assignment of the Supplicant and Authenticator roles (for example, in peer-to-peer IEEE 802.11 networking) or where both systems desire to make use of their own Authentication Server for controlling access to their services. These concepts are further discussed in 6.7.

6.5 Reception and transmission control

The degree to which protocol exchanges that take place on the controlled Port are affected by the authorization state is determined by two *controlled directions* parameters associated with each controlled port: an *AdminControlledDirections* parameter and an *OperationalControlledDirections* parameter. These parameters determine whether a controlled Port that is unauthorized exerts control over communication in both directions (disabling both reception of incoming frames and transmission of outgoing frames), or just in the incoming direction (disabling only reception of incoming frames). The controlled directions parameters can take one of two possible values, *Both* and *In*. The relationship between these two parameters, and the meaning of their values, is as follows:

a) **AdminControlledDirections = Both.** This indicates that control is required to be exerted over both incoming and outgoing traffic through the controlled Port. The value of OperControlledDirections is unconditionally set equal to Both if AdminControlledDirections is set equal to Both.

b) **AdminControlledDirections = In.** This indicates that control is required to be exerted only over incoming traffic through the controlled Port. If AdminControlledDirections is set equal to In, the value of OperControlledDirections is set equal to In on initialization and when the Port's MAC becomes operable. However, the value of OperControlledDirections is set to Both if any of the following conditions is true:

   1) The Port is a Bridge Port, and the Bridge Detection state machine (see Clause 17 of IEEE Std 802.1D) detects the presence of another Bridge connected to the Port.

   2) The Port is a Bridge Port, and the Edge Port parameter for the Port is FALSE.

   3) The Port's MAC becomes inoperable.

The value of the AdminControlledDirections parameter can only be modified by management. Implementations of Port-based access control shall support the ability to independently set the AdminControlledDirections parameter for each controlled Port to the value Both. Implementations of Port-based access control may support the ability to independently set the AdminControlledDirections parameter for each controlled Port to the value In.

NOTE—The In setting allows the security provisions of Port-based access control to be relaxed on Ports where an attached device needs to see protocol traffic from a controlled Port to support some forms of startup and initialization function (e.g., Wake-on-LAN), but where it is still desirable to prevent the attached device from transmitting into the controlled Port until authentication has taken place (see B.1). Clearly, the relaxation of the control offered by the controlled Port in this way reduces the effectiveness of Port-based access control in dealing with some types of attack.

The value of OperControlledDirections is determined by the operation of the Controlled Directions state machine (see 8.2.10).

6.6 Port Access Entity (PAE)

A Port Access Entity (PAE) operates the algorithms and protocols associated with the Port Access Control Protocol defined in Clause 8. A PAE exists for each Port of a System that supports Port Access Control functionality in the Supplicant role, the Authenticator role, or both.

In the Supplicant role, a PAE is responsible for providing information to an Authenticator that will establish its credentials. A PAE that performs the Supplicant role in an authentication exchange is known as a Supplicant PAE.

In the Authenticator role, a PAE is responsible for communication with a Supplicant, and for submitting the information received from the Supplicant to a suitable Authentication Server in order for the credentials to be checked and for the consequent authorization state to be determined. A PAE that performs the Authenticator role in an authentication exchange is known as an Authenticator PAE.

Both PAE roles control the authorized/unauthorized state of the controlled Port (see 6.4) depending on the outcome of the authentication process. If a given controlled Port has both Authenticator PAE and Supplicant PAE functionality associated with it, both PAEs must be in the Authorized state in order for the controlled Port to become Authorized.

Subclauses 6.6.1 through 6.6.4 offer an overview of the operation of a PAE in the Authenticator and Supplicant roles.

6.6.1 Authenticator role

An Authenticator PAE is responsible for enforcing the authentication of a Supplicant PAE that attaches to its controlled Port and for controlling the authorization state of the controlled Port accordingly.

In order to perform the authentication, the Authenticator PAE makes use of an Authentication Server. The Authentication Server may be co-located in the same System as the Authenticator PAE, or it may be located elsewhere, accessible via remote communication mechanisms, LAN-based or otherwise. Communication between the Supplicant PAE and the Authenticator PAE, and between the Authenticator PAE and the Authentication Server (when the Authentication Server is not co-located with the Authenticator), is achieved by means of the protocols and procedures described in Clause 8.

6.6.2 Supplicant role

A Supplicant PAE is responsible for communicating the credentials of the Supplicant to the Authenticator PAE in response to requests from the Authenticator PAE, and for controlling the authorization state of the controlled Port according to the outcome of the authentication exchange communicated to it by the Authenticator PAE. The Supplicant PAE may also initiate authentication exchanges and perform explicit logoff exchanges, as further described in 6.6.4 and Clause 8.

6.6.3 Port access restrictions

Authentication occurs primarily at System initialization time, or when a Supplicant System is connected to a Port of an Authenticator System. Until authentication has successfully completed, the Supplicant System only has access to the Authenticator System to perform authentication exchanges, or to access any services offered by the Authenticator's System that are not subject to the access control restrictions placed on either the Authenticator's controlled Port or the Supplicant's controlled Port (see 6.4 and 6.5). Once authentication has successfully completed, both Systems can allow full access by the Supplicant System to the services offered via the Authenticator System's controlled Port.

The operational state of the MAC that supports a controlled Port can be disabled or enabled. If the MAC's operational state is disabled, then the MAC is not available for use, regardless of the authorization state associated with the controlled Port.

NOTE 1—Clause 6 of IEEE Std 802.1D describes the parameters available in a Bridge Port that indicate the enable/disable states of the Port's MAC. IEEE Std 802.3 describes similar parameters that define the enable/disable states of the logical MAC offered by an aggregation.

In a System that implements a PAE, the controlled Port is placed in the unauthorized state until authentication has taken place, and it is therefore also disabled. Once authentication has succeeded, and it has been determined that the authenticated user is authorized to access the controlled Port, the controlled Port is placed in the authorized state; assuming that there is no other reason for it still to be disabled (e.g., the MAC has been disabled for administrative purposes), the controlled Port is then available for use (see 6.4).

NOTE 2—While an Authenticator System's controlled Port is in the unauthorized state, Dynamic Host Configuration Protocol (DHCP) and other initialization traffic may not be transmitted or received via the controlled Port, depending on the current value of the Port's Controlled Directions parameters (see 6.5). As a result, authentication may need to occur early in the end-station initialization sequence [prior to DHCP and Internet Protocol (IP) initialization for example].
In addition, while the Supplicant's controlled Port is in the unauthorized state, transmission or reception of DHCP and other initialization traffic is not possible. Attempts by the Supplicant to utilize its port prior to authorization may result in loss of DHCP or initialization packets, potentially resulting in an inability to obtain an IP address.

In addition to controlling the controlled Port's authorization state, the operation of the PAE may support ageing out of the authorization state of controlled Ports, and it may request that the Supplicant reauthenticate at any time. Controlled Ports remain authorized during reauthentication and transition to the unauthorized state only if reauthentication fails.

Authentication is configurable on a per-Port basis, because it will be desirable in some configurations not to perform authentication on certain Ports (e.g., on inter-Bridge links, Ports attached to servers). The management operations available for such configuration are described in Clause 9.

6.6.4 Logoff mechanisms

There are several mechanisms that can result in the controlled Port state changing to unauthorized and thereby controlling access via that Port in accordance with its OperControlledDirections parameter (see 6.5):

a) The authentication exchanges between the Supplicant and the Authentication Server can result in failure to authorize the Port.

b) Management controls can prevent the Port from being authorized, regardless of the credentials of the Supplicant.

c) The MAC associated with the Port can be non-operational for any reason (including for hardware failure or administrative reasons).

d) Connection failure between the Supplicant and the Authenticator can result in the Authenticator timing out the authorization state.

e) Expiry of a reauthentication timer can occur without successful reauthorization.

f) The Supplicant PAE can fail to respond to a request for authentication information by the Authenticator PAE.

g) The Supplicant PAE can issue an explicit logoff request.

When a user logs off from an end station, it is possible in some environments for the user (or a different user) to bypass a new login request and thereby gain access to the end station and the network. Providing the explicit logoff mechanism ensures that the session is terminated, not only with respect to the user's access to the end station, but also for the end station's authorization status with the controlled Port of the Authenticator System to which it is connected. An explicit logoff therefore causes both the Supplicant and Authenticator PAEs to set their controlled Ports to the unauthorized state.

6.7 Coupling two EAPOL authentications

This subclause discusses the ability of the Port Access Control Protocol (PACP) specified in Clause 8 to transport two simultaneous authentication dialogs in opposite directions, how implementation asymmetries affect authentication results, and how simultaneous bi-directional transport may be utilized.

In the PACP, there is a requestor role (Authenticator) and a responder role (Supplicant). The Authenticator transmits frames to the Supplicant and the Supplicant responds to those frames with frames of its own. The PACP, like the authentication protocol carried within it (EAP), is a request/response protocol, and the state machines reflect this. Such a transport is sometimes called "unidirectional" meaning the protocol exchanges always originate from one side (the requestor).

However, it is not the IEEE 802.1X transport that controls whether the authentication is mutual or one-way. Rather, the chosen EAP method controls whether the authentication is mutual or one-way, and support for a Supplicant controlled port determines whether authorization is mutual or one-way.

Despite the asymmetry of IEEE 802.1X and EAP transports, if a mutually authenticating EAP method [such as EAP-Transport Layer Security (EAP-TLS)] is carried within IEEE 802.1X frames, then the Supplicant and Authenticator can mutually authenticate. However, if an EAP method is chosen that only authenticates the Supplicant to the Authenticator (e.g., EAP MD5-CHALLENGE), then the authentication will be one-way.

In an earlier version of IEEE Std 802.1X (see IEEE Std 802.1X-2001 [B6]), only the Authenticator was specified to have a controlled port. The lack of a Supplicant controlled port prevented the Supplicant from enforcing authentication decisions, thus creating a possibility of Supplicant connecting to a rogue

Authenticator even when a mutual authentication method was used. However, this was due to the lack of a specification of the controlled port on the Supplicant, not due to the one-way nature of the PACP transport protocol. The current version of IEEE Std 802.1X remedies this by specifying how both Authenticator and Supplicant roles affect the controlled port.

In the PACP, each role has its own state machine and it is possible that a single device can implement both the Supplicant and Authenticator roles (state machines). If a device implementing both roles encounters another device implementing both roles, then two separate (and simultaneous) PACP transports will take place in opposite directions. This is sometimes referred to as a bi-directional authentication transport or two coupled one-way authentication exchanges. PACP does not support "tie breaking" wherein two hosts initiating authentication with each other will only go forward with a single authentication.

Two coupled one-way authentication exchanges are not equivalent in security to a single exchange of a mutually authenticating EAP method (such as EAP-TLS). Since the two coupled one-way authentication exchanges are not cryptographically bound together, there is no way to ascertain that the party involved in the first one-way exchange is the same party that is involved in the alternate one-way exchange.

Even though a single PACP exchange may provide mutual authentication, there may be other reasons to run two PACP exchanges in opposite directions. RFC 3748, Section 2.4 discusses some of these cases, which include:

a) When the devices require the creation of separate key material in each direction but the keying protocol is uni-directional (as in the group handshake in IEEE 802.11 adhoc networks).

b) When different credentials are used for different roles (one credential for the Supplicant role and one for the Authenticator).

c) When two bridges are connected, each bridge may implement the Supplicant role, but may have authorization requirements that can only be enforced by the Authenticator ("Supplicant Access Control with Authenticator" variable set to inactive).

When a device that implements both PACP roles on a single port encounters a device that implements only a single Authenticator role, this may result in asymmetric controlled port state (the single role device authorizes its controlled port while the dual role device does not because its Authenticator role has not been satisfied). A complete table of encounters and results appears in Table 6-1.

It is suggested that applications utilizing IEEE Std 802.1X should specify which PACP roles must be implemented to avoid encounters that will result in failed network connections. See Section 2.4 of RFC 3748 for other issues that may arise in peer-to-peer usage of EAP.

6.8 Use of Port Access Control with IEEE Std 802.3

Where Port Access Control is used in conjunction with Ports that can be aggregated in accordance with Clause 43 of IEEE Std 802.3, the authentication and authorization mechanisms of the PAE shall operate on the individual Ports, and Ports shall be considered not to be aggregatable while they are in the Unauthorized state. Any Port that has become part of an aggregation, and that subsequently becomes Unauthorized, shall therefore be removed from that aggregation.

NOTE 1—The above is a requirement for interoperability in LAN environments that support both Clause 43 of IEEE Std 802.3 and Port Access Control. The background to this requirement is discussed in C.1.2.

Table 6-1—Result of encounters between EAPOL transport roles assuming key-generating EAP methods are run and authentication result is success.

		Local		
		Supplicant	**Authenticator**	**Both**
Remote	Supplicant	*Fails gracefully* No link will be formed if media is considered unsafe without encryption (IEEE Std 802.11). Unauthenticated link will be formed if media is considered safe by default (IEEE Std 802.3).	*Works* Authenticated link. Authentication policy is set by choice of EAP method.	*Works* Authenticated link. Authentication policy set by choice of EAP method. Remote Supplicant and local Authenticator are authorized. Local Supplicant will receive no response and assume no Authenticator. Successful authentication to the local Authenticator will have made the port secure.
	Authenticator	*Works* Authenticated link. Authentication policy is set by choice of EAP method.	*Fails gracefully* No link will be formed. Each Authenticator will send initial EAP Requests but no response will arrive.	*Fails* Asymmetric link. Remote controlled port is authorized, but local controlled port remains unauthorized. Remote Authenticator and local Supplicant are authorized, local Authenticator remains unauthorized.
	Both	*Works* Authenticated link. Authentication policy set by choice of EAP method. Local Supplicant and remote Authenticator are authorized. Remote Supplicant will receive no response and assume no Authenticator. Successful authentication to the remote Authenticator will have made the port secure.	*Fails* Asymmetric link. Local controlled port is authorized, but remote controlled port remains unauthorized. Local Authenticator and remote Supplicant are authorized, remote Authenticator remains unauthorized.	*Works* Authenticated link. Authentication policy set by choice of two uncoupled EAP methods. Two keys will be formed and some mechanism must exist so that local and remote choose the same key.

In order to ensure the secure creation of an aggregation in this environment, the Ports that form a single aggregation should be authorized, either to the same entity, or to entities that are all authorized to join the aggregation. The process of determining if a Port is actually authorized to join an aggregation is beyond the scope of this standard, but could be determined by the Authentication Server and returned to the Authenticator as part of a successful authentication exchange. For example, in the case of RADIUS, an aggregation key could be generated using the attributes returned with the Access-Accept message.

NOTE 2—Link Aggregation will combine this aggregation key with other aggregation key information in order to create the final aggregation key to be used in LACP.

7. EAP encapsulation over LANs (EAPOL)

This clause defines the encapsulation techniques that shall be used to carry EAP packets between Supplicant PAEs and Authenticator PAEs in a LAN environment. The encapsulation is known as *EAP over LANs*, or *EAPOL*. At present, EAPOL encapsulations are described for IEEE 802.3/Ethernet MACs and Token Ring/FDDI MACs. The EAPOL encapsulation used with IEEE 802.3/Ethernet MACs can be applied to other LAN technologies that share the same basic frame format as Ethernet (for example, IEEE 802.12 Demand Priority operating in IEEE 802.3 compatibility mode). Similarly, the EAP encapsulation used with Token Ring/FDDI MACs can be applied to other LAN technologies that share the same basic frame format as IEEE 802.5 Token Ring (for example, FDDI or IEEE 802.12 Demand Priority operating in IEEE 802.5 compatibility mode).

NOTE—In this standard, "IEEE 802.3/Ethernet MACs" is used to refer to IEEE 802 MACs in which the native Link Layer protocol identification mechanism is based on a choice between the Type interpretation and Length interpretation of the Length/Type field. "Token Ring/FDDI MACs" is used to refer to MACs in which the native Link Layer protocol identification mechanism is based on LLC addressing.

7.1 Transmission and representation of octets

All EAPOL PDUs consist of an integral number of octets, numbered starting from 1 and increasing in the order that they are put into a MAC frame. The bits in each octet are numbered from 1 to 8, where 1 is the low-order bit.

When consecutive octets are used to represent a binary number, the lower numbered octet contains the more significant bits of the binary number.

When the encoding of (an element of) an EAPOL PDU is represented using a diagram in this clause, the following representations are used:

a) Octet 1 is shown toward the top of the page, higher numbered octets being toward the bottom.

b) Where more than one octet appears on a given line, octets are shown with the lowest numbered octet to the left, higher numbered octets being to the right.

c) Within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

7.2 EAPOL MPDU format for use with IEEE 802.3/Ethernet

A summary of the Ethernet form of an EAPOL MPDU is shown in Figure 7-1, starting with the Length/Type field of the MPDU. The fields shown in the diagram are defined in 7.5.

	Octet Number
PAE Ethernet Type (7.5.1)	1-2
Protocol Version (7.5.3)	3
Packet Type (7.5.4)	4
Packet Body Length (7.5.5)	5-6
Packet Body (7.5.6)	7-N

Figure 7-1—EAPOL MPDU format for IEEE 802.3/Ethernet

7.3 EAPOL MPDU format for use with IEEE 802.2 Logical Link Control (LLC)

A summary of the form of IEEE 802.2 LLC form of an EAPOL MPDU is shown in Figure 7-2. The fields shown in the diagram are defined in 7.5.

	Octet Number
SNAP-encoded Ethernet Type (7.5.2)	1-8
Protocol Version (7.5.3)	9
Packet Type (7.5.4)	10
Packet Body Length (7.5.5)	11-12
Packet Body (7.5.6)	13-N

Figure 7-2—EAPOL MPDU format for IEEE 802.2 LLC

7.4 Tagging EAPOL MPDUs

EAPOL MPDUs transmitted by a PAE shall not be Virtual LAN (VLAN) tagged, but may optionally be priority tagged. All PAEs shall be capable of receiving both priority tagged and untagged EAPOL MPDUs.

The structure of the tag header used for priority tagging is specified in IEEE Std 802.1Q.

7.5 EAPOL MPDU field and parameter definitions

7.5.1 PAE Ethernet type

This field is two octets in length, and it shall contain the Ethernet Type value assigned for use by the PAE, as defined in Table 7-3.

7.5.2 Subnetwork Access Protocol (SNAP)-encoded Ethernet type

This field is eight octets in length, and it shall contain the SNAP-encoded Ethernet Type, encoded in SNAP format, as follows:

a) Octets numbered 1 through 3 carry the standard SNAP header, consisting of the hexadecimal value AA-AA-03.

b) Octets numbered 4 through 6 carry the SNAP Protocol Identifier (PID), consisting of the hexadecimal value 00-00-00.

c) Octets 7 and 8 carry the PAE Ethernet Type value, as defined in Table 7-3.

7.5.3 Protocol version

This field is one octet in length, taken to represent an unsigned binary number. Its value identifies the version of EAPOL protocol supported by the sender of the EAPOL frame. An implementation conforming to this specification shall use the value 0000 0002 in this field.

7.5.4 Packet type

This field is one octet in length, taken to represent an unsigned binary number. Its value determines the type of packet being transmitted. The following types are defined:

a) **EAP-Packet.** A value of 0000 0000 indicates that the frame carries an EAP packet.

b) **EAPOL-Start.** A value of 0000 0001 indicates that the frame is an EAPOL-Start frame.

c) **EAPOL-Logoff.** A value of 0000 0010 indicates that the frame is an explicit EAPOL-Logoff request frame.

d) **EAPOL-Key.** A value of 0000 0011 indicates that the frame is an EAPOL-Key frame.

e) **EAPOL-Encapsulated-ASF-Alert.** A value of 0000 0100 indicates that the frame carries an EAPOL-Encapsulated-ASF-Alert.

All other possible values of this field shall not be used, as they are reserved for use in potential future extensions to this protocol.

NOTE—The EAPOL-Key packet type is used only where the optional ability to transmit key information between the Authenticator and the Supplicant is supported (see 8.1.9).

The EAPOL-Encapsulated-ASF-Alert packet type is provided for use by the Alerting Standards Forum (ASF) as a means of allowing alerts (e.g., specific SNMP traps; see B.1.4) to be forwarded through a Port that is in the Unauthorized state. All EAPOL frames with this packet type that are received on the uncontrolled Port are passed to the protocol entity responsible for handling ASF alerts for validation and further processing in accordance with the relevant ASF protocol specifications. This standard does not further specify either the syntax or semantics of the alert messages that can be carried in this type of packet, or the protocol actions taken on receipt of a packet of this type.

7.5.5 Packet Body length

This field is two octets in length, taken to represent an unsigned binary number. The value of this field defines the length in octets of the Packet Body field (see 7.5.6); a value of 0 indicates that there is no Packet Body field present.

7.5.6 Packet Body

The Packet Body field is present if the Packet Type contains the value EAP-Packet, EAPOL-Key, or EAPOL-Encapsulated-ASF-Alert; for all other values of Packet Type, this field is not present.

In a frame carrying a Packet Type of EAP-Packet, this field contains an EAP packet as described in 7.7. Exactly one EAP packet is encapsulated.

In a frame carrying a Packet Type of EAPOL-Key, this field contains a Key Descriptor as described in 7.6. Exactly one Key Descriptor is encapsulated.

In a frame carrying a Packet Type of EAPOL-Encapsulated-ASF-Alert, this field contains an ASF alert frame as specified by the ASF (see B.1.4). Exactly one ASF alert frame is encapsulated.

NOTE—The maximum size of EAP packet that can be carried within an EAPOL frame will depend on the maximum MAC frame size supported by the MAC method by which the frame is transmitted.

7.5.7 Validation of received EAPOL MPDUs and EAPOL protocol version handling

A PAE shall process a received EAPOL MPDU as specified in this clause if and only if the following conditions are all true:

a) The Destination MAC address field contains the PAE group address, as specified in 7.8 (in non-shared media LANs), or the specific MAC address of the PAE (in shared media LANs).

b) The PAE Ethernet Type field contains the value of the PAE Ethernet Type, as specified in 7.8.

c) The Packet Type field contains one of the values EAP-Packet, EAPOL-Start, EAPOL-Logoff, or EAPOL-Key, as specified in 7.5.4.

NOTE 1—The model of operation, as described in 6.4, is such that any MPDU received by the underlying MAC is made available at both the Controlled Port and the Uncontrolled Port. The Port Access Entity is always attached to the Uncontrolled Port (and never attached to the Controller Port); hence, regardless of the state of the Controlled Port, any EAPOL MPDU that is made available at the Controlled Port is discarded.

In the case of EAPOL MPDUs that carry the Packet Type values EAPOL-Start and EAPOL-Logoff, any octets that appear in the PDU following the Packet Type field shall be ignored. In the case of EAPOL MPDUs that carry the Packet Type value EAP-Packet and EAPOL-Key, any octets that appear in the PDU following the Packet Body field shall be ignored.

The following rules apply to the validation and interpretation of EAPOL MPDUs, in order to ensure that backward compatibility is maintained between versions of this protocol.

For an implementation that supports version A of the protocol, a received EAPOL MPDU of a given Packet Type that carries a protocol version number B is interpreted as follows:

d) Where B is greater than or equal to A, the EAPOL PDU shall be interpreted as if it carried the supported version number, A. Specifically:

   1) All EAPOL PDU parameters that are defined in version A shall be interpreted in the manner specified for version A of the protocol for the given EAPOL PDU Packet Type.

   2) All EAPOL PDU parameters that are undefined in version A for the given EAPOL PDU Packet Type shall be ignored.

   3) All octets that appear in the EAPOL PDU beyond the largest numbered octet defined for version A for the given EAPOL PDU Packet Type shall be ignored.

NOTE 2—As a consequence of the rules stated in item d) and its sub-bullets, a version 1 implementation may ignore the Protocol Version Identifier. This set of validation rules allows the possibility of future specification of extensions to the EAPOL Protocol, identified as new versions by different values of the identifier. Subsequent versions will be required to check the Protocol Version Identifier in order to correctly interpret the received PDU.

e) Where B is less than A, the EAPOL PDU shall be interpreted as specified for the version number, B, carried in the EAPOL PDU. Specifically:

   1) All EAPOL PDU parameters shall be interpreted in the manner specified for version B of the protocol for the given EAPOL PDU Packet Type.

   2) All EAPOL PDU parameters that are undefined in version B for the given EAPOL PDU Packet Type shall be ignored.

   3) All octets that appear in the EAPOL PDU beyond the largest numbered octet defined for version B for the given EAPOL PDU Packet Type shall be ignored.

7.6 Key Descriptor format

A summary of the Key Descriptor format is shown in Figure 7-3. The fields shown in the diagram are defined in the following subclauses.

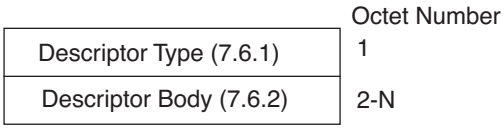


Figure 7-3—Key Descriptor format

7.6.1 Descriptor type

This field is one octet in length, taken to represent an unsigned binary number. The value defines the type of of the Key Descriptor, which in turn defines how the Descriptor Body is used and interpreted.

Table 7-2 lists the values of Descriptor Type that are currently assigned.

Table 7-2—Descriptor Type value assignments

Assignment	Value
RC4 Key Descriptor Type (see 7.6.3)[a]	1
IEEE 802.11 Key Descriptor Type (see 7.6.4)[b]	2

[a]NOTE—Use of this descriptor type is deprecated—see D.5.7.
[b]IEEE 802.11 Key Descriptor Type has been assigned for use by the IEEE 802.11 standard.

All other possible values of Descriptor Type are reserved for future standardization.

As the Descriptor Type number space is small and finite, the first two fields in the Descriptor Body (see 7.6.2) shall comprise a Subtype and a Version,[9] thus allowing revisions of, and extensions to, existing Descriptor Types to be defined without incurring further depletion of Descriptor Type values. The format of the Subtype and Version fields is not constrained by this standard, and forms part of the definition of the descriptor Body associated with each Descriptor Type.

7.6.2 Descriptor Body

The Descriptor Body contains an integral number of octets. The format and semantics of the Descriptor Body is as defined for a particular value of Descriptor Type.

[9]The RC4 Descriptor Type is the only exception to this rule, as it was already defined at the time the rule was imposed.

7.6.3 RC4 Key Descriptor

Use of the RC4® Key Descriptor[10] is now deprecated (see D.5.7); however, its definition is included here for consistency with earlier versions of this standard.

NOTE—Annex D contains a more detailed description of the RC4 descriptor type.

The format of the RC4 Key Descriptor is shown in Figure 7-4. The Descriptor Type takes the value 1 to indicate an RC4 Key Descriptor; the remaining fields are as defined in the subclauses that follow.

	Octet Number
Descriptor Type = 1 (7.6.1)	1
Key Length (7.6.3.1)	2-3
Replay Counter (7.6.3.2)	4-11
Key IV (7.6.3.3)	12-27
Key Index (7.6.3.4)	28
Key Message Digest (7.6.3.5)	29-44
Key (7.6.3.6)	45-Packet Body Length

Figure 7-4—RC4 Key Descriptor format

7.6.3.1 Key length

This field is two octets in length, taken to represent an unsigned binary number. The value defines the length of the key in octets. For example, a value of 5 in this field indicates a 40-bit key.

7.6.3.2 Replay counter

This field is 8 octets in length, taken to represent an unsigned binary number. It carries a counter value, used to detect and prevent replay of key messages.

7.6.3.3 Key IV

This field carries a 16-octet Initialization Vector value, consisting of 128 bits of random data.

7.6.3.4 Key index

This field is one octet in length, taken to represent a 7-bit unsigned binary number and a flag. The value is generated by the Authenticator specifying the key, and it is used as a key index number if multiple keys are supported. The index number is carried in bits 1 through 7, and it can carry an integer in the range 0–127. Bit 8 is a flag bit. If bit 8 is set to 1, the key is a unicast key; if bit 8 is set to 0, the key is a broadcast key.

[10]The mark RC4 is a registered trademark of RSA Security Inc. and may not be used by third parties creating implementations of the algorithm. RSA Security does not hold any patents nor does it have any pending applications on the RC4 algorithm. However, RSA Security does not represent or warrant that implementations of the algorithm will not infringe the intellectual property rights of any third party. Proprietary implementations of the RC4 encryption algorithm are available under license from RSA Security Inc. For licensing information, contact RSA Security Inc., 2955 Campus Drive, Suite 400, San Mateo, CA 94403-2507, USA, or http://www.rsasecurity.com.

7.6.3.5 Key message digest

This field is 16 octets in length. The Key Message Digest is a message digest of all of the fields of the EAPOL packet, from and including the EAPOL protocol version field, to and including the Encrypted Key field, with the message digest set to 0. The Key Message Digest is an HMAC-MD5 message digest over the EAPOL MPDU, as described in 7.6.3.7.

7.6.3.6 Key

This field is optional. If it is not present, the Supplicant uses the peer key generated as part of the EAP authentication process as the key material for this message. If the key is longer than the key length specified in this message, then only the first N octets are used, where N is equal to the value of key length.

When RADIUS is used as the Authentication Server, and when the MS-MPPE-Send-Key and MS-MPPE-Recev-Key (see IETF RFC 3079 and IETF RFC 2548) are used to transport cryptographic keys, then the key field is interpreted as follows:

If Packet Body Length = 44 + Key Length, then the Key Field contains the key in encrypted form, of length Key Length. If Packet Body Length = 44, then the Key field is absent, and the least significant Key Length octets from the MS-MPPE-Send-Key attribute are used as the keying material. Where the Key field is encrypted using RC4, the RC4 encryption key used to encrypt this field is formed by concatenating the 16 octet (128 bit) Key-IV field with the 32 octet MS-MPPE-Recv-Key attribute. This yields a 48 octet RC4 key (384 bits).

7.6.3.7 Construction and interpretation of an RC4 Key Descriptor

Key Descriptors carrying a Descriptor Type of RC4 Key Descriptor are constructed and interpreted as follows:

a) The Replay Counter field carries an NTP time value (see IETF RFC 1305).

b) The Key IV field carries a random number used to generate an RC4 encryption key.

c) A message digest type of HMAC-MD5 is used to generate the Key Message Digest (see IETF RFC 2104). The key used for the message digest is the server key generated by the EAP authentication (e.g., as defined in IETF RFC 2716).

d) RC4 is used to encrypt the Key field. The RC4 encryption key is generated by concatenating the Key IV and the session key generated by the EAP authentication process (e.g., as defined in IETF RFC 2716). The key material is then encrypted according to the method specified by Encrypt Type (e.g., RC4 encrypted using the RC4 key).

7.6.4 IEEE 802.11 Key Descriptor Type

A single value of Key Descriptor Type has been assigned for use by IEEE Std 802.11, as shown in Table 7-2. The format and semantics of the Descriptor Body for these Key Descriptor Types are as defined in IEEE Std 802.11.[11] Authenticator and Supplicant key machines utilizing the IEEE 802.11 key descriptor are also defined by IEEE Std 802.11; these state machines completely replace the machines defined in this document when IEEE Std 802.1X is used in conjunction with the IEEE 802.11 key exchange mechanism.

[11] At the time of publication of this standard, the specification of the IEEE 802.11 Key Descriptor was “work in progress” as part of the development of P802.11i.

7.7 EAP packet format—informative

A summary of the EAP packet format is shown in Figure 7-5. The figure and the accompanying field descriptions are included for illustrative purposes only; the normative definition of the EAP packet format and its associated semantics are to be found in the specification of EAP contained in IETF RFC 3748.

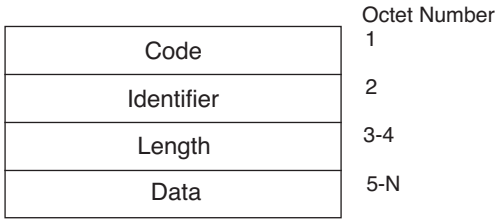


Figure 7-5—EAP packet format

7.7.1 Code

The Code field is one octet in length and identifies the type of EAP packet. EAP Codes are assigned as follows:

1 Request
2 Response
3 Success
4 Failure

7.7.2 Identifier

The Identifier field is one octet in length and allows matching of responses with requests. The Code field, Identifier field, and System Port together uniquely identify an authentication exchange. The Authentication server must also maintain a separate identifier space for each session, and the System Port identifier should be used for this purpose.

The operation of EAP (see 8.1.1) in the Authenticator determines the value of Identifier used in new EAP-Request frames. The Supplicant uses the same Identifier in subsequent EAP-Response frames responding to that initial request frame. The Authenticator PAE uses the same Identifier value in any retransmissions of the same request.

7.7.3 Length

The Length field is two octets in length and indicates the length of the EAP packet, including the Code, Identifier, Length, and Data fields.

NOTE—Because in this specification EAP frames are transported directly over the link layer medium, the frame size cannot exceed the maximum permissible on that medium, as fragmentation is not supported.

7.7.4 Data

The Data field is zero or more octets. The format of the Data field is determined by the Code field.

7.8 EAPOL addressing

PAEs (see Clause 8) transmit and receive:

a) EAPOL frames exchanged between an Authenticator and a Supplicant

b) EAP frames exchanged between an Authenticator and an Authentication Server

NOTE 1—An Authentication Server may be co-located with an Authenticator; in which case, a communication protocol may not be required.

NOTE 2—This standard refers to EAP carried in RADIUS as a basis of these exchanges; however, the use of other protocols to achieve these exchanges is permitted.

PAEs use the DL_UNITDATA.request and DL_UNITDATA.indication primitives associated with each active Port to transmit and receive EAPOL frames. In IEEE 802.3/Ethernet MACs, frames carry the Port Access Entity Ethernet Type as the data link address; this type value is defined in Table 7-3. In Token Ring/FDDI MACs, frames carry the Port Access Entity Ethernet Type in SNAP-encoded form, using a SNAP header of AA-AA-03 and a SNAP PID of 00-00-00, followed by the Ethernet Type value.

Table 7-3—Standard Ethernet Type assignment

Assignment	Value
Port Access Entity Ethernet Type	88-8E

A group MAC address, the PAE group address (see Table 7-4), is assigned by IEEE Std 802.1D[12] for use by PAEs. In MACs where the LAN technology concerned is such that the individual MAC address of the Supplicant is known to the Authenticator, and vice versa (for example, in IEEE 802.11, where the establishment of an association between a station and an access point involves the exchange of MAC addresses), all EAPOL frames transmitted by a PAE shall carry the individual MAC address associated with the destination PAE's point of LAN attachment as the destination MAC address. Otherwise (i.e., in MACs where the LAN technology concerned is such that the individual MAC address of the Supplicant can be unknown to the Authenticator, and vice versa, for example IEEE Std 802.3), all EAPOL frames transmitted by a PAE shall carry the PAE group address as the destination MAC address even if the individual MAC address of the destination PAE is later discovered. All EAPOL frames shall carry the individual MAC address associated with the source PAE's point of LAN attachment to its counterpart (Authenticator or Supplicant PAE) as the source MAC address. In an IEEE 802.11 access point, the BSSID associated with the wireless interface would be used in place of this MAC address.

Table 7-4—PAE group address assignment

Assignment	Value
Port Access Entity group address (PAE group address)	01-80-C2-00-00-03

NOTE—The PAE group address is one of the reserved set of group MAC addresses that are not forwarded by MAC Bridges.

[12]The assignment of this address is to be found in Table 7-9 of IEEE Std 802.1D. The value is repeated here for informative purposes.

7.9 Use of EAPOL in shared media LANs

In order to make use of EAPOL in shared media LANs, it is necessary to establish a pairwise association between the Supplicant and Authenticator. The use of individual MAC addresses with EAPOL (7.8) permits such an association to be established, and in particular, this has been allowed in order to support the use of Port-based Network Access Control in IEEE 802.11 wireless LAN networks. However, it should be noted that in shared media LANs, it is necessary to take steps to ensure:

a) Reliability of the authentication that takes place between Authenticator and Supplicant.

b) Data confidentiality for data transmitted on the association subsequent to successful authentication.

This standard takes no steps to provide either reliable authentication or data confidentiality in a shared medium environment. The use of EAPOL in a shared medium environment renders Port-based network access control highly vulnerable to attack. In addition, use of EAPOL in an environment where reliable transmission is not supported will result in frequent timeouts; even though EAP supports retransmission, (see RFC 3748, Section 4.3) frequent timeouts will result in a degraded user experience.

8. Port Access Control Protocol

8.1 Introduction to protocol operation

8.1.1 Overview

The operation of the authentication process makes use of the Extensible Authentication Protocol (EAP, specified in IETF RFC 3748) as the means of communicating authentication information between the Supplicant and the Authentication Server. EAP is a general protocol that supports multiple authentication mechanisms. For example, through the use of EAP, support for a number of authentication schemes may be added, including smart cards, Kerberos, Public Key Encryption, One Time Passwords, and others.

The approach taken in this standard is to define an encapsulation format that allows EAP Messages to be carried directly by a LAN MAC service. The encapsulated form of EAP, known as EAP over LANs, or EAPOL, is used for all communication between the Supplicant PAE and the Authenticator PAE.

Each PAE has two separate components, a set of PACP state machines, and a higher layer with which these machines communicate. In the case of the Supplicant PAE, the higher layer consists of EAP functionality, while in the case of the Authenticator PAE, the higher layer is a combination of EAP and authentication, authorization, and accounting (AAA) functionality. This standard defines the PACP state machines and the interface between the PACP state machines and the higher-layer functionality.

The operation of the higher-layer functions with which the PAE state machines communicate is outside the scope of this standard. EAP protocol exchanges are defined by IETF EAP standards, IETF RFC 3748, and successor standards. One example of a AAA protocol, RADIUS, is defined by IETF RADIUS standards, IETF RFC 2865, IETF RFC 2866, IETF RFC 3579, and successor standards. Annex F defines the interface that the PACP PAE state machines assume from the higher layers.

Figure 8-1 shows the interface between the PACP state machines and the higher layer for the Supplicant PAE and Authenticator PAE. As shown, the portEnabled signal from the system indicates to both the higher layer and the PACP that a port is active. The PACP passes EAP messages between the physical port and the higher layer. Message flow on the Supplicant is controlled using eapResp/eapNoResp from EAP to indicate it is ready for another message and eapReq from the PACP to indicate a message is available for EAP to process. Message flow on the Authenticator side is controlled with a similar process, with the higher layer using eapReq/eapNoReq, to indicate when it is ready to receive a new message, and eapResp to indicate that a message is available to be processed by the higher layer.

Within the higher layer, EAP and associated EAP methods drive the authentication dialog, but on completion the higher layer will take its cue from AAA to signal success or failure to the PACP, using eapSuccess and eapFail signals. The separation between the higher layer and the PACP is such that all EAP messages exchanged between Supplicant and Authenticator are created by the EAP component.

The EAP method on the Authenticator acts as a “passthrough,” forwarding EAP messages back and forth to an Authentication Server using a AAA protocol as a transport. The higher layer coordinates the EAP and AAA components. RADIUS is one such transport protocol that can be used as a lower layer for this forwarding mechanism. The use of RADIUS for pass-through forwarding is described in IETF RFC 3579.

Rather than only permitting a predetermined authentication method, EAP allows the Authenticator PAE to request more information before determining the specific authentication mechanism. In EAP, the Authenticator PAE sends one or more Requests to authenticate the Supplicant PAE. The Request has a type field to indicate what is being requested. Examples of Request types include Identity, MD5-challenge, One-Time Passwords, and Generic Token Card. The MD5-challenge type corresponds closely to the Challenge Handshake Authentication Protocol (CHAP). Typically, the Authenticator will send an initial Identity Request

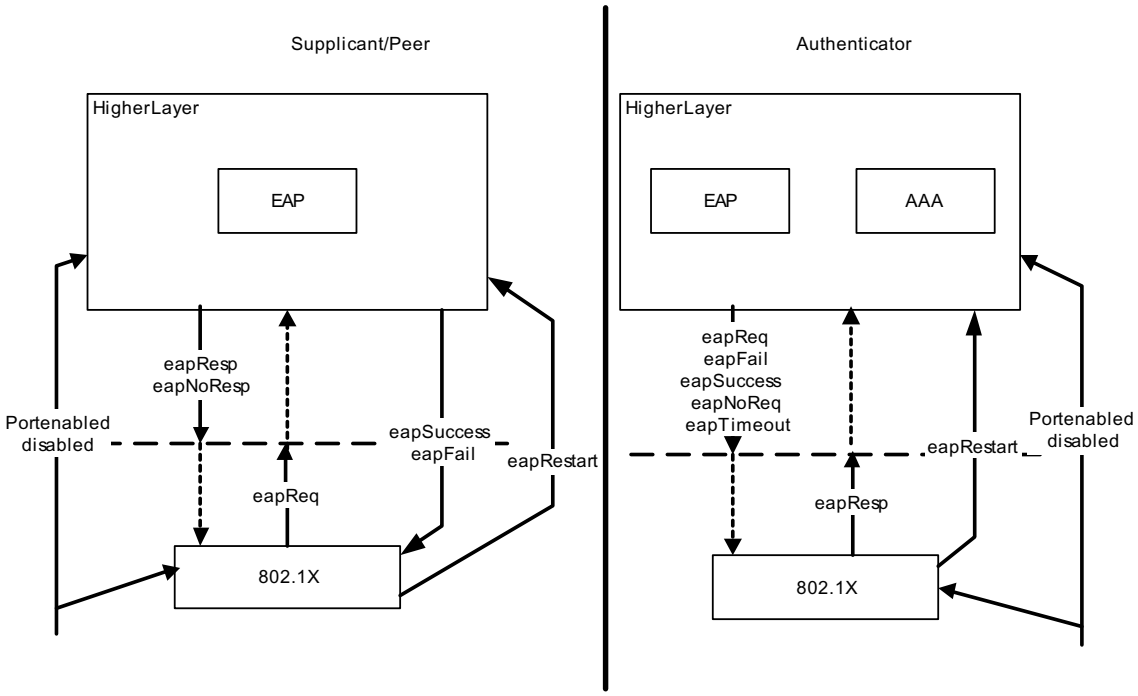


Figure 8-1—Higher layer interface diagram

followed by one or more Requests for authentication information. However, an initial Identity Request is not required, and it may be bypassed in cases in which the identity is presumed or can be determined by other means (such as a method-specific identity exchange). The Supplicant PAE sends a Response packet in reply to each Request. As with the request packet, the Response packet contains a type field that corresponds to the type field of the Request.

The authentication exchange ends with an Accept or Reject indication from the Authentication Server. The Authenticator forwards the EAP packet included within the Accept or Reject indication, while the AAA client operating as a lower layer beneath EAP interprets the indication and indicates success or failure to the Authenticator, which will set the controlled Port to authorized or unauthorized appropriately.

8.1.2 Authentication initiation

Authentication can be initiated either by the Supplicant PAE or by the Authenticator PAE. If authentication is enabled on a given Port (i.e., if the Port supports an Authenticator PAE and the AuthControlledPortControl parameter for the Port is set to Auto and SystemAuthControl is set to Enabled; see 6.4), authentication is initiated by the Authenticator PAE on sensing that the operational state of the MAC associated with the Port has transitioned from disabled to enabled. As noted below, if the Authenticator PAE does not receive a response, EAP will retransmit the authentication request.

A Supplicant PAE may initiate the authentication sequence by sending an EAPOL-Start frame (see 7.5.4).

NOTE 1—On initialization, a Supplicant PAE may not be listening for an authentication initiation during the early portions of the initialization sequence, and consequently, the Supplicant PAE may miss the initiation frame(s) sent by the Authenticator PAE. As a result, if the Supplicant PAE did not initiate authentication itself, an authentication failure could occur purely due to timing issues. A Supplicant PAE that does not receive an authentication initiation frame from the Authenticator PAE on reinitialization may initiate authentication by sending an EAPOL-Start frame.

NOTE 2—A previously authenticated Supplicant PAE's System may be reinitialized. In this circumstance, an Authenticator PAE might not initiate authentication, because it may not sense a transition in the MAC's operational state. In this case, it could be argued that authentication is not necessary because the Supplicant PAE has already been authenticated.

However, it is possible in some environments to reinitialize a machine, bypass the normal login, and access the Authenticator System's services. To prevent an unauthorized user from accessing the Authenticator by reinitializing an authenticated machine, a Supplicant initiation of authentication is necessary upon reinitialization.

8.1.2.1 Authenticator initiation

The Authenticator PAE will typically initiate the conversation when it receives an indication that the Port has become operable. Before authentication commences, the Port state is forced to the unauthorized state.

The Authenticator PAE initiates the authentication sequence by signaling the higher layer and then sending the EAP-Request frame given to it by EAP. Typically, EAP will begin the authentication exchange with an EAP-Request frame; however, any EAP-Request can be used to initiate the exchange. A Supplicant PAE receiving an EAP-Request frame from the Authenticator PAE responds with an EAP-Response frame provided by EAP.

Authenticator PAEs may support periodic reauthentication, and they may request that a Port reauthenticate at any time. For example, if the Authenticator System reinitializes, the authentication state can be recovered by issuing EAP-Request frames on all Ports. If a controlled Port is in the authorized state prior to reauthentication, then it will remain in that state during reauthentication. If the authentication fails for a controlled Port that was in the authorized state during reauthentication, then the controlled Port's authorization state is transitioned to unauthorized in order to control external access to that Port in accordance with the current value of the OperControlledDirections parameter (see 6.5).

8.1.2.2 Supplicant initiation

In order to request that the Authenticator PAE initiate authentication, the Supplicant PAE sends an EAPOL-Start packet (see 7.5.4). The Authenticator PAE receiving an EAPOL-Start packet responds by sending an EAP-Request packet that is chosen by EAP.

8.1.3 EAPOL-Logoff

When a Supplicant wishes the Authenticator PAE to perform a logoff (i.e., to set the controlled Port state to unauthorized), the Supplicant PAE originates an EAPOL-Logoff message (see 7.5.4) to the Authenticator PAE. As a result, the Authenticator PAE immediately places the controlled Port in the unauthorized state.

NOTE—In general, it is advisable for the Supplicant PAE to originate an EAPOL-Logoff in any circumstances in which the authentication credentials are user-based, and the user of the Supplicant System has logged off (in the case of an end station), or in which the operation of the Supplicant System has been reconfigured in a manner that would invalidate any previous authentication results (for example, a management change that affects the Supplicant System's identity, or its authorization to use the services of the Authenticator's System).

8.1.4 Timing out authorization state information

Authenticator PAEs can time out the authorization state information on a periodic basis by means of the Reauthentication Timer State Machine (see 8.2.8). The time period for such timeouts is reAuthPeriod seconds since the last time that the authorization state was confirmed. The state variable reAuthEnabled controls whether periodic reauthentication takes place.

Reauthentication can be enabled and disabled, and the reAuthPeriod modified, by management. The default settings are for the reAuthPeriod to be 3600 s (one hour) and for reauthentication to be disabled.

NOTE—As with Authenticator and Supplicant initiated reauthentication, the implications of setting this to a lower value should be carefully thought out before proceeding. The value chosen will be affected by the reliability with which the MAC associated with the Port can detect and indicate MAC enabled/disabled conditions. If the Port's detection of MAC state is reliable, then longer timeout values may be appropriate.

8.1.5 Retransmission

As noted in the specification of EAP, the Authenticator PAE is responsible for retransmission of messages between the Supplicant PAE and the Authenticator PAE. In particular, EAP is the Authenticator PAE component that handles retransmission, not the IEEE 802.1X state machines. Thus, if an EAP-Packet is lost in transit between the Supplicant PAE and the Authenticator PAE (or vice versa), the Authenticator PAE will retransmit. The exceptions are EAPOL-Start messages, which are retransmitted, if necessary by the Supplicant PAE, and any EAP messages delivered during the FAIL and SUCCESS states. Because the messages delivered from the Authenticator FAIL and SUCCESS states (typically EAP-Failure and EAP-Success) are not acknowledged by the Supplicant PAE, they are not retransmitted by the Authenticator PAE. If an EAP-request is lost, the Supplicant PAE state machine transitions to the CONNECTING state on authWhile timer expiration.

In implementations in which the authentication function is performed by a remote Authentication Server, retransmissions may be necessary between the Authenticator PAE and the Authentication Server. In this instance, it may be necessary for the higher layer to adopt a retransmission strategy that is more appropriate to the transmission characteristics of the communication path involved.

NOTE—DHCP clients incorporate a randomized exponential backoff algorithm to determine the delay between retransmissions, allowing a retransmission delay of up to 64 s. Assuming that the Authenticator is in blocking mode prior to authentication and initiates authentication on receiving an initial DHCP frame from the end station, the initial DHCP frame will be dropped and will trigger authentication initiation by the Authenticator. Assuming that the authentication can complete in time to avoid a DHCP timeout, the DHCP conversation will complete successfully.

It may be necessary to adjust retransmission strategies and authentication timeouts in certain cases. For example, when a token card is used, additional time may be required to allow the user to find the card and enter the token. If the user takes a particularly long time to find the card, then a DHCP timeout can occur. This problem cannot be ameliorated by enqueing the initial DHCP frames because DHCP client timers are started when the packets are enqueued, not when they are sent.

8.1.6 Migration considerations

It is desirable that the transition between a non-authenticated and an authenticated environment be as smooth as possible. For example, when an authentication-capable Supplicant connects to a non-authentication-capable Bridge, the Supplicant will not receive an EAP-Request packet. As a result, the Supplicant PAE will initiate an EAPOL-Start frame to the PAE group MAC address. As this address is one of the addresses that are not forwarded by MAC Bridges, the Supplicant PAE will not receive a response. Hence, after a suitable timeout period in which the EAPOL-Start has been retransmitted and no response has been received, the Supplicant can assume that it is authorized to access the Bridged Local Area Network.

As the concept of the controlled and uncontrolled Ports applies to systems that support either Authenticator or Supplicant functionality, a system that only supports Supplicant functionality cannot transmit data frames prior to completion of the authentication process. Hence, in the example above, the Supplicant system must wait for the end of the authentication timeout period before initiating data exchanges via the Bridge Port. In order to allow applications such as DHCP to run immediately on connection, it may be desirable to administratively force the Supplicant Port to the Authorized state in such situations.

NOTE—If the Supplicant finds that the authentication process times out, the choice as to whether the Supplicant enables the controlled Port or not is determined by the security policy in force in the Supplicant system.

When a non-authentication-aware Supplicant connects to an authentication enabled Bridge, the Supplicant, having no PAE, will ignore EAP-Request frames. Consequently, the Port will remain in the unauthorized state. The Supplicant will be able to access the Bridged Local Area Network via the Bridge's controlled Port only in accordance with the value of the OperControlledDirections parameter (see 6.5); access to any services that are made available via the Bridge's uncontrolled Port are unrestricted.

8.1.7 Relaying EAP frames

The Authenticator PAE is responsible for relaying EAP frames between the Supplicant and the Authentication Server via the higher layer. It must also perform any repackaging of EAP frames that is necessary in order to convert EAP frames carried as EAPOL between Supplicant PAE and Authenticator PAE. It is the responsibility of the higher layer to perform any repackaging of EAP frames required between the Authenticator and Authentication Server. In performing its relay function, the information contained in the EAP frames relayed is not modified other than as required to convert the frame format to/from the EAPOL format.

NOTE—EAP in RADIUS (see IETF RFC 2865, IETF RFC 2866, IETF RFC 3579, and Annex D) offers a suitable option for the AAA protocol used between Authenticator and Authentication Server; however, use of RADIUS is not mandated by this standard.

EAPOL-Start and EAPOL-Logoff frames are transmitted by the Supplicant PAE to the Authenticator PAE; EAPOL-Key frames are transmitted by the Authenticator PAE to the Supplicant PAE and by the Supplicant PAE to the Authenticator PAE. These frames are not relayed onward by the Authenticator PAE to the Authentication Server.

The initial EAP-Request frame is typically transmitted by the Authenticator PAE to the Supplicant PAE and may not appear in the communication path between the Authentication Server and the Authenticator PAE.

All EAP frames received from the Supplicant PAE are decapsulated by the Authenticator PAE from their EAPOL format for transfer as EAP frames to EAP. It is the higher layer's responsibility to format the EAP frame for onward transmission to the Authentication Server in accordance with the AAA protocol in use between the Authentication Server and the Authenticator PAE.

All AAA protocol frames received by the Authenticator PAE from the Authentication Server are converted to EAP frames by the higher layer and then passed to the Authenticator PAE. The Authenticator PAE then converts these EAP frames to EAPOL format, as appropriate for the Port concerned, for onward transmission to the Supplicant PAE.

8.1.8 Example EAP exchanges

In these example exchanges, the exchange of EAPOL frames is shown as a solid line; the exchange of EAP frames carried in a higher-layer protocol such as RADIUS is shown by a broken line. The diagrams therefore show which PDU exchanges involve the Authenticator PAE in repackaging EAP frames in order to perform protocol translation.

NOTE 1—For the purposes of these examples, Figure 8-2 through Figure 8-7 show the Authenticator PAE and higher layer as a single entity under the heading "AUTHENTICATOR PAE". In the diagrams, the communications shown between the headings "SUPPLICANT PAE" and "AUTHENTICATOR PAE" are managed by the Authenticator PAE and the communications shown between the headings "AUTHENTICATOR PAE" and the "AUTHENTICATION SERVER" are managed by the higher layer.

NOTE 2—For the purposes of these examples, it has been assumed that the AAA client indicates success to the Authenticator when an Accept message is received from the Authentication Server, and that the AAA client indicates failure to the Authenticator when a Reject message is received from the Authentication Server. However, it is possible for EAP to indicate success or failure to the Supplicant based on whatever criteria the EAP specification allows and would typically provide EAP-Success or EAP-Failure frames to be delivered to the Supplicant (although this is not required). Because the Backend Authentication state machine is influenced only by the success or failure indication from the AAA client, rather than the encapsulated EAP packet, this does not present a problem.

The example in Figure 8-2 shows an Authenticator-initiated conversation for the case of a One Time Password (OTP) authentication. OTP is used only for illustrative purposes; other authentication protocols could also have been used, although they may show somewhat different behavior.

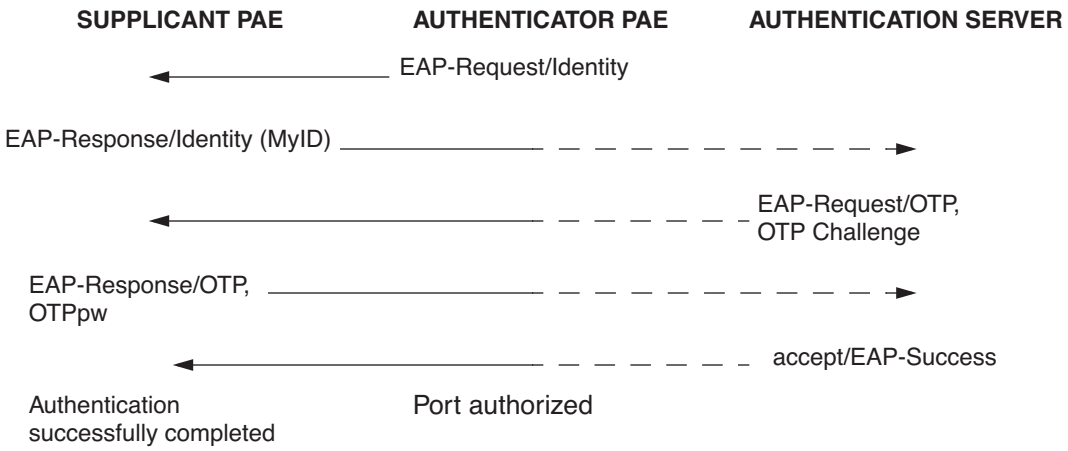


Figure 8-2—Authenticator-initiated, one-time password exchange (success)

In the case in which the Supplicant fails EAP authentication, the conversation would appear as illustrated in Figure 8-3.

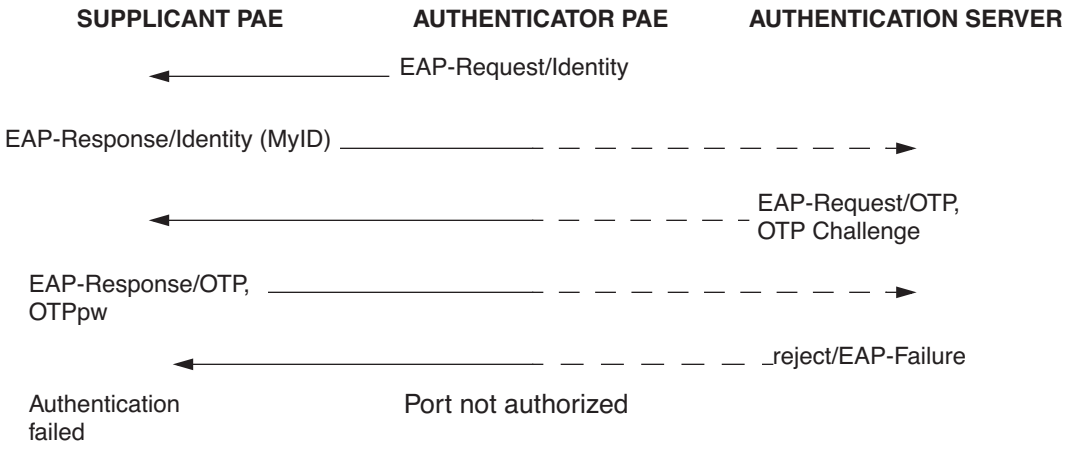


Figure 8-3—Authenticator-initiated, one-time password exchange (failure)

Figure 8-4 illustrates a successful authentication exchange, followed by an explicit logoff requested by the Supplicant.

A Supplicant-initiated authentication conversation will appear as illustrated in Figure 8-5.

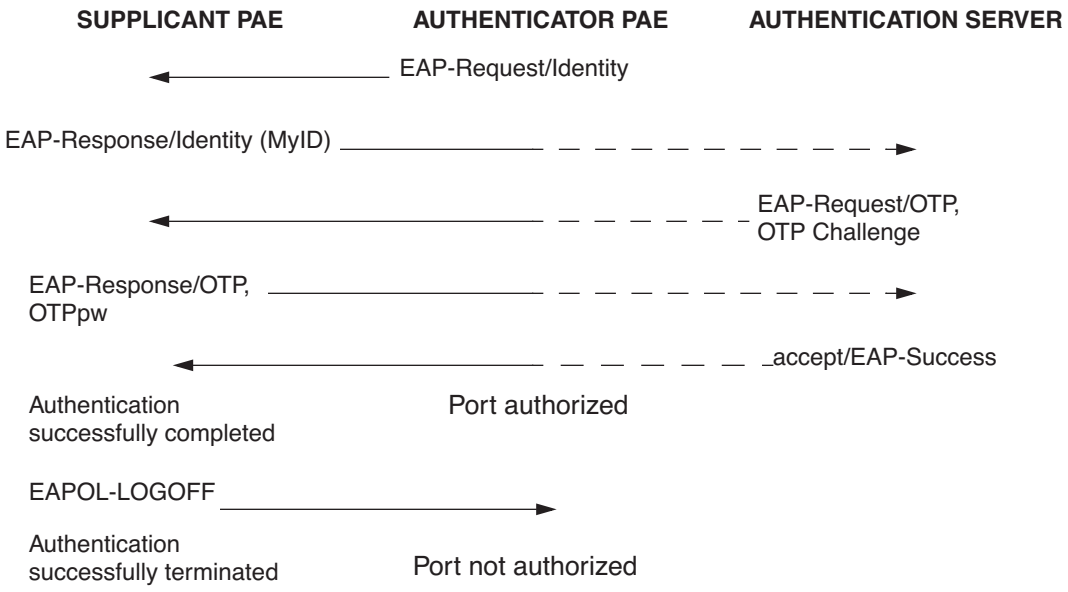


Figure 8-4—Successful authentication followed by Supplicant-initiated logoff

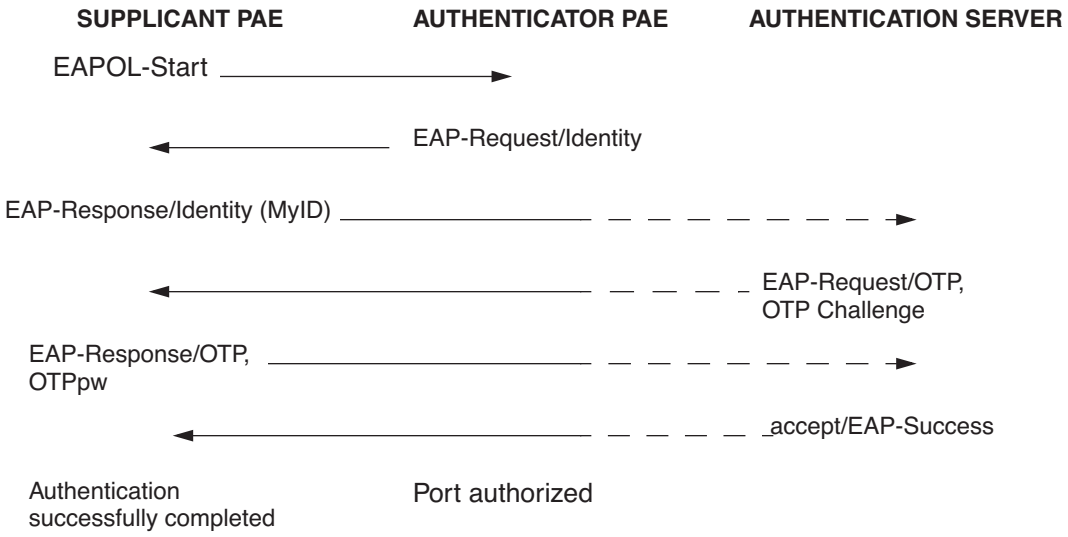


Figure 8-5—Supplicant-initiated, one-time password exchange (success)

In the case in which the Supplicant does not support authentication, but authentication is enabled on that Authenticator, the conversation would appear as illustrated in Figure 8-6.

In the case in which the Authenticator does not support authentication, but authentication is enabled on that Supplicant, the conversation would appear as illustrated in Figure 8-7.

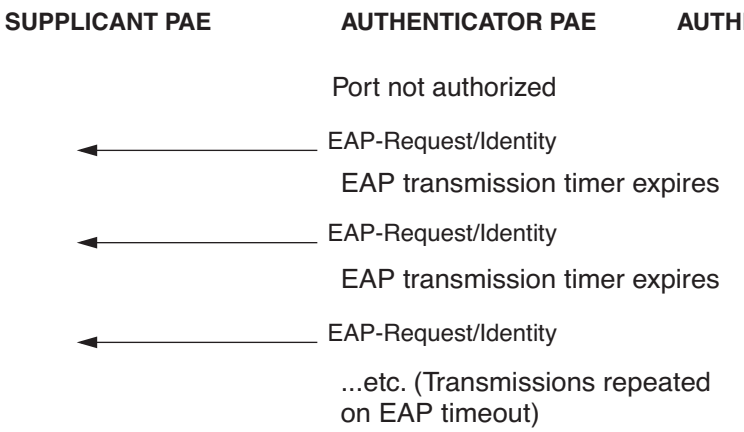


Figure 8-6—Supplicant does not support authentication

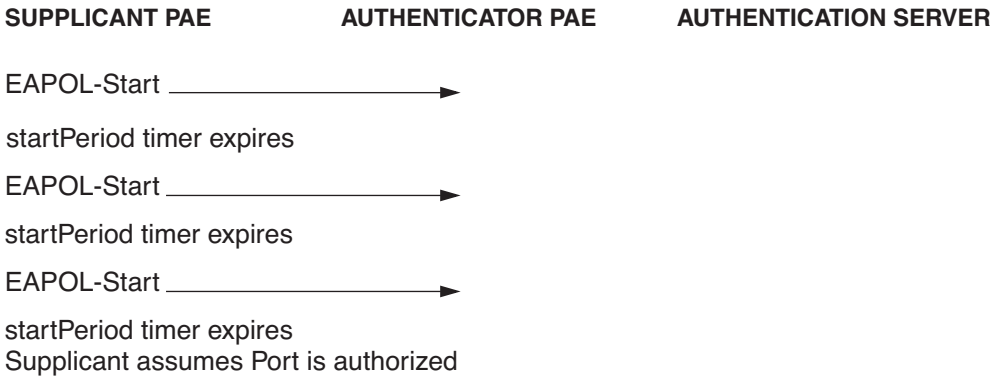


Figure 8-7—Authenticator does not support authentication

8.1.9 Transmission of key information

The EAPOL protocol optionally supports the transmission of key information from the Authenticator to the Supplicant, or from the Supplicant to the Authenticator, following a successful authentication exchange, in circumstances in which encryption is available between the Supplicant and Authenticator systems (e.g., where encryption is used on an IEEE 802.11 association between a station and an access point).

The use of this facility is controlled by the keyTxEnabled parameter, which may be modified by management. A value of TRUE allows key information to be transmitted once keyAvailable and keyRun are set TRUE.

keyAvailable may be set TRUE administratively, or may be set by the higher layer during authentication. Figure 8-8 shows the interface between the higher layer and the IEEE 802.1X Key layer that supports transmitting key material and signaling keyAvailable. Note that if keyTxEnabled is TRUE then EAP authentication dialog must result in keys being available at both Authenticator and Supplicant for this to be successful, so only EAP methods that support this can be used in environments where EAP is expected to provide the keys. The PACP also supports a variable portValid that is available to the higher layer (this is not used by

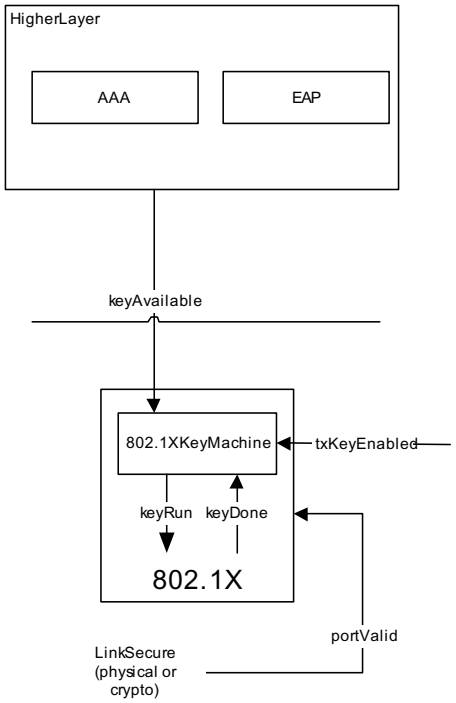


Figure 8-8—Key machine higher layer

existing implementations but is available for future implementations). portValid influences when an authenticated port becomes authorized. This allows the PACP to require two conditions before authorizing a port: Authenticated and data channel secure (e.g., encrypted).

NOTE—This standard provides a mechanism for transmitting key information following authentication; however, it does not specify for what reason such information is transmitted. The decision as to which key values are transmitted is made externally to the operation of the Authenticator and Supplicant and their associated state machines.

8.2 PACP state machines

The following PACP state machines are described:

a) The Port Timers state machine (see 8.2.3)

b) The Authenticator PAE state machine (see 8.2.4)

c) The Authenticator Key Transmit state machine (see 8.2.5)

d) The Supplicant Key Transmit state machine (see 8.2.6)

e) The Key Receive state machine (see 8.2.7)

f) The Reauthentication Timer state machine (see 8.2.8)

g) The Backend Authentication state machine (see 8.2.9)

h) The Controlled Directions state machine (see 8.2.10)

i) The Supplicant PAE state machine (see 8.2.11)

j) The Supplicant Backend state machine (see 8.2.12).

These state machines are defined on a per-Port basis. Table 8-1 summarizes the state machine support requirements for implementations that support Authenticator functionality, Supplicant functionality, or both.

An X marked in a cell of the table indicates that the State Machine named on that row of the table shall be supported in an implementation that claims to support the functionality identified for the column concerned. An O marked in a cell of the table indicates that the State Machine named on that row of the table may be supported in an implementation that claims to support the functionality identified for the column concerned.

Table 8-1—State machine support requirements

	Functionality Supported		
State Machine Name	**Authenticator**	**Supplicant**	**Both**
Port Timers state machine (see 8.2.3)	X	X	X
Authenticator PAE state machine (see 8.2.4)	X		X
The Authenticator Key Transmit state machine (see 8.2.5)	O		O
The Supplicant Key Transmit state machine (see 8.2.6)		O	O
Reauthentication Timer state machine (see 8.2.8)	X		X
Backend Authentication state machine (see 8.2.9)	X		X
Controlled Directions state machine (see 8.2.10)	X		X
Supplicant PAE state machine (see 8.2.11)		X	X
Supplicant Backend State Machine (see 8.2.12)		X	X
The Key Receive state machine (see 8.2.7)	X	X	X

8.2.1 Notational conventions used in state diagrams

State diagrams are used to represent the operation of the protocol by a number of cooperating state machines each comprising a group of connected, mutually exclusive states. Only one state of each machine can be active at any given time.

Each state is represented in the state diagram as a rectangular box, divided into two parts by a horizontal line. The upper part contains the state identifier, written in upper case letters. The lower part contains any procedures that are executed on entry to the state.

All permissible transitions between states are represented by arrows, the arrowhead denoting the direction of the possible transition. Labels attached to arrows denote the condition(s) that must be met in order for the transition to take place. All conditions are expressions that evaluate to TRUE or FALSE; if a condition evaluates to TRUE, then the condition is met. The label UCT denotes an unconditional transition (i.e., UCT always evaluates to TRUE). A transition that is global in nature (i.e., a transition that occurs from any of the possible states if the condition attached to the arrow is met) is denoted by an open arrow; i.e., no specific state is identified as the origin of the transition. When the condition associated with a global transition is met, it supersedes all other exit conditions including UCT. The special global condition BEGIN supersedes all other global conditions, and once asserted remains asserted until all state blocks have executed to the point that variable assignments and other consequences of their execution remain unchanged.

Table 8-2—State machine symbols

Symbol	Interpretation
( )	Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes.
;	Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal English language conventions for reading text.
=	Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., a = b = X the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator.
!	Logical NOT operator.
&&	Logical AND operator.
||	Logical OR operator.
if...then...	Conditional action. If the Boolean expression following the **if** evaluates to TRUE, then the action following the **then** is executed.
{statement 1, ... statement N}	Compound statement. Braces are used to group statements that are executed together as if they were a single statement.
!=	Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.
==	Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right.
<	Less than. Evaluates to TRUE if the value of the expression to the left of the operator is less than the value of the expression to the right.
>	Greater than. Evaluates to TRUE if the value of the expression to the left of the operator is greater than the value of the expression to the right.
>=	Greater than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either greater than or equal to the value of the expression to the right.
+	Arithmetic addition operator.
–	Arithmetic subtraction operator.

On entry to a state, the procedures defined for the state (if any) are executed exactly once, in the order that they appear on the page. Each action is deemed to be atomic; i.e., execution of a procedure completes before the next sequential procedure starts to execute. No procedures execute outside of a state block. The procedures in only one state block execute at a time, even if the conditions for execution of state blocks in different state machines are satisfied, and all procedures in an executing state block complete execution before the transition to and execution of any other state block occurs, i.e., the execution of any state block appears to be atomic with respect to the execution of any other state block and the transition condition to that state from the previous state is TRUE when execution commences. The order of execution of state blocks in different state machines is undefined except as constrained by their transition conditions. A variable that is set to a particular value in a state block retains this value until a subsequent state block executes a procedure that modifies the value.

On completion of all of the procedures within a state, all exit conditions for the state (including all conditions associated with global transitions) are evaluated continuously until one of the conditions is met. The label ELSE denotes a transition that occurs if none of the other conditions for transitions from the state are met (i.e., ELSE evaluates to TRUE if all other possible exit conditions from the state evaluate to FALSE). Where two or more exit conditions with the same level of precedence become TRUE simultaneously, the choice as to which exit condition causes the state transition to take place is arbitrary.

Where it is necessary to split a state machine description across more than one diagram, a transition between two states that appear on different diagrams is represented by an exit arrow drawn with dashed lines, plus a reference to the diagram that contains the destination state. Similarly, dashed arrows and a dashed state box are used on the destination diagram to show the transition to the destination state. In a state machine that has been split in this way, any global transitions that can cause entry to states defined in one of the diagrams are deemed to be potential exit conditions for all of the states of the state machine, regardless of which diagram the state boxes appear in.

Should a conflict exist between the interpretation of a state diagram and either the corresponding global transition tables or the textual description associated with the state machine, the state diagram takes precedence. The interpretation of the special symbols and operators used in the state diagrams is as defined in Table 8-2; these symbols and operators are derived from the notation of the "C++" programming language, ISO/IEC 14882 [B7]. If a Boolean variable is described in this clause as being set it has or is assigned the value TRUE, if reset or clear the value FALSE.

8.2.2 Timers and global variables used in the definition of the state machines

8.2.2.1 Timers

The timers defined for these state machines are decremented, if their value is nonzero, by the operation of the Port Timers state machine (see 8.2.3). All timers used by the PACP state machines have a resolution of one second; i.e., the initial values used to start the timers are integer values, and they represent the timer period as an integral number of seconds.

NOTE 1—It is permissible to introduce a degree of jitter into the initialization of these timers; for example, in order to distribute the timing of EAPOL frame transmissions among Ports in multi-Port implementations.

NOTE 2—Correct operation of the protocol depends upon the use of compatible initialization values for the authWhile timer in the Supplicant and the timer used by EAP for request retransmission in the Authenticator. As there is no automatic means of communicating changes in timer values between Authenticator and Supplicant, deviation from the default initialization values for these timers can adversely affect the operation of the protocol.

a) **authWhile.** A timer used by the Supplicant PAE to determine how long to wait for a request from the Authenticator before timing it out. The initial value of this timer is authPeriod.

b) **aWhile.** A timer used by the Backend Authentication state machine in order to determine timeout conditions in the exchanges between the Authenticator and EAP. The initial value of this timer is serverTimeout.

c) **heldWhile.** A timer used by the Supplicant state machine to define periods of time during which it will not attempt to acquire an Authenticator. The initial value of this timer is heldPeriod.

d) **quietWhile.** A timer used by the Authenticator state machine to define periods of time during which it will not attempt to acquire a Supplicant. The initial value of this timer is quietPeriod.

e) **reAuthWhen.** A timer used by the Reauthentication Timer state machine to determine when reauthentication of the Supplicant takes place. The initial value of this timer is reAuthPeriod.

f) **startWhen.** A timer used by the Supplicant PAE state machine to determine when an EAPOL-Start PDU is to be transmitted. The initial value of this timer is startPeriod.

8.2.2.2 Global variables

Global variables are available for use by more than one state machine and are used to perform interstate-machine communication and initialization functions.

a) **authAbort.** This variable is set TRUE by the Authenticator PAE state machine in order to signal to the Backend Authentication state machine to abort its authentication procedure. Its value is set FALSE by the Backend Authentication state machine once the authentication procedure has been aborted.

b) **authFail.** This variable is set TRUE if the authentication process (represented by the Backend Authentication state machine) fails. It is set FALSE by the operation of the Authenticator PAE state machine, prior to initiating authentication.

c) **authPortStatus.** The current authorization state of the Authenticator PAE state machine. This variable is set to Unauthorized or Authorized by the operation of the state machine. If the Authenticator PAE state machine is not implemented, then this variable has the value Authorized.

d) **authStart.** This variable is set TRUE by the Authenticator PAE state machine in order to signal to the Backend Authentication state machine to start its authentication procedure. Its value is set FALSE by the Backend Authentication state machine once the authentication procedure has been started.

e) **authTimeout.** This variable is set TRUE if the authentication process (represented by the Backend Authentication state machine) fails to obtain a response from the Supplicant. The variable may be set by management action, or by the operation of a timeout while in the AUTHENTICATED state. This variable is set FALSE by the operation of the Authenticator PAE state machine.

f) **authSuccess.** This variable is set TRUE if the authentication process (represented by the Backend Authentication state machine) succeeds. It is set FALSE by the operation of the Authenticator PAE state machine, prior to initiating authentication.

g) **eapFail.** This variable is set TRUE by the higher layer if it determines that the authentication has failed.

h) **eapolEap.** This variable is set TRUE by an external entity if an EAPOL PDU carrying a Packet Type of EAP-Packet is received.

i) **eapSuccess.** This variable is set TRUE by the higher layer if it determines that the authentication has been successful.

j) **eapTimeout.** This variable is set TRUE by the higher layer if it determines that the Supplicant is not responding to requests.

k) **initialize.** This variable is externally controlled. When asserted, it forces all EAPOL state machines to their initial state. The PACP state machines are held in their initial state until initialize is deasserted.

l) **keyAvailable.** Set to TRUE by an external entity when there is a new key available that can be used by the key machines to begin a new key exchange. Set FALSE when the Key Transmit state machines have transmitted the key value.

m) **keyDone.** This variable is set TRUE by the key machine when it is in a state that portValid can be tested.

n) **keyRun.** This variable is set TRUE by the PACP machine when the transmit key machine should run. It is set FALSE by a PAE to indicate the PAE state machine has been reset and the key machine should abort.

o) **keyTxEnabled.** Reflects the current value of the KeyTransmissionEnabled parameter (see 8.1.9).

NOTE 1—The value of keyTxEnabled is a constant, in the sense that the state machines do not modify its value; however, its value may be modified by management.

p) **portControl.** This variable is derived from the current values of the *AuthControlledPortControl* and *SystemAuthControl* parameter (see 6.4) for the Port. This variable can take the following values:

1) **ForceUnauthorized.** The controlled Port is required to be held in the Unauthorized state.

2) **ForceAuthorized.** The controlled Port is required to be held in the Authorized state.

3) **Auto.** The controlled Port is set to the Authorized or Unauthorized state in accordance with the outcome of an authentication exchange between the Supplicant and the Authentication Server.

If SystemAuthControl is set to Enabled, then portControl directly reflects the value of the AuthControlledPortControl parameter. If SystemAuthControl is set to Disabled, then the value of portControl is ForceAuthorized.

q) **portEnabled.** This variable is externally controlled. Its value reflects the operational state of the MAC service supporting the Port. Its value is TRUE if the MAC service supporting the Port is in an operable condition (see 6.4), and it is otherwise FALSE.

r) **portStatus.** The current authorization state of the controlled Port. This variable is set to Unauthorized or Authorized by the operation of the PAE state machines. The value of portStatus directly determines the value of the AuthControlledPortStatus parameter (see 6.4) for the Port.

The value of portStatus is determined from the values of authPortStatus, suppPortStatus, and the *Supplicant Access Control With Authenticator* administrative control parameter (see 6.4), as follows:

1) If both Supplicant PAE and Authenticator PAE state machines are implemented for the Port, and the value of the Supplicant Access Control With Authenticator administrative control parameter is *inactive*, then the value of portStatus directly reflects the value of authPortStatus. Otherwise:
2) If the value of either authPortStatus or suppPortStatus is Unauthorized, then the value of portStatus is Unauthorized. Otherwise:
3) If the values of authPortStatus and suppPortStatus are both Authorized, then the value of portStatus is Authorized.

s) **portValid.** This variable is TRUE if:

1) The Port technology provides a level of security such that it is acceptable to assume that the Port is secure without requiring additional means to protect the traffic across the Port—for example, where the Port is the single point of attachment offered by a single LAN MAC in a Bridged Local Area Network infrastructure; or
2) The Port requires specific actions to take place in order for a secure channel to be established between Supplicant and Authenticator, and those actions have successfully taken place—for example, in an IEEE 802.11 LAN, an end station has associated to an access point, and keys have been exchanged in order to allow traffic on that association to be encrypted.

The value of this variable is otherwise FALSE.

The value of this variable is used by the PACP state machines; however, its value is determined by procedures and protocols that are external to the PACP state machines, and that are outside the scope of this standard.

NOTE 2—This variable is used in conjunction with keyDone. The potential combinations determine the meaning of portValid:

— portValid and keyDone are both TRUE: The port is in a valid state.
— portValid is TRUE and keyDone is FALSE: The port is not in a valid state; it is in an unknown state because the key machines have not run yet.

— portValid is FALSE and keyDone is TRUE: The port is not in a valid state due to the key protocol failing.

— portValid and keyDone are both FALSE: The port is not in a valid state; it is an unknown state because the key machines have not run yet.

When portValid is FALSE it only indicates a key exchange failure if keyDone is true; otherwise, it indicates that the port is not valid for some other reason.

t) **reAuthenticate.** This variable is set TRUE by the Reauthentication Timer state machine on expiry of the reAuthWhen timer. This variable may also be set TRUE by management action. It is set FALSE by the operation of the Authenticator PAE state machine. Reauthentication may not begin immediately. The Authenticator does not interrupt the current authentication, but instead waits for it to complete before beginning a new authentication. Only one pending reauthentication will be tracked.

u) **suppAbort.** This variable is set TRUE by the Supplicant PAE state machine to signal to the Supplicant Backend state machine that it should abort an authentication sequence. It is set FALSE by the Supplicant Backend state machine.

v) **suppFail.** This variable is set FALSE by the Supplicant PAE state machine, and is set TRUE by the Supplicant Backend state machine on completion of an unsuccessful authentication sequence.

w) **suppPortStatus.** The current authorization state of the Supplicant PAE state machine. This variable is set to Unauthorized or Authorized by the operation of the state machine. If the Supplicant PAE state machine is not implemented, then this variable has the value Authorized.

x) **suppStart.** This variable is set TRUE by the Supplicant PAE state machine to signal to the Supplicant Backend state machine that it should start an authentication sequence. It is set FALSE by the Supplicant Backend state machine.

y) **suppSuccess.** This variable is set FALSE by the Supplicant PAE state machine, and is set TRUE by the Supplicant Backend state machine on completion of a successful authentication sequence.

z) **suppTimeout.** This variable is set FALSE by the Supplicant PAE, and is set TRUE by the Supplicant Backend state machine if the authentication sequence times out.

8.2.3 Port Timers state machine

The Port Timers state machine for a given Port is responsible for decrementing the timer variables for that Port each second, in response to an external system clock function. The timer variables are used, and set to their initial values, by the operation of the individual state machines for the Port.

The Port Timers state machine shall implement the function specified by the state diagram contained in Figure 8-9 and the attendant timer definitions contained in 8.2.2.1.

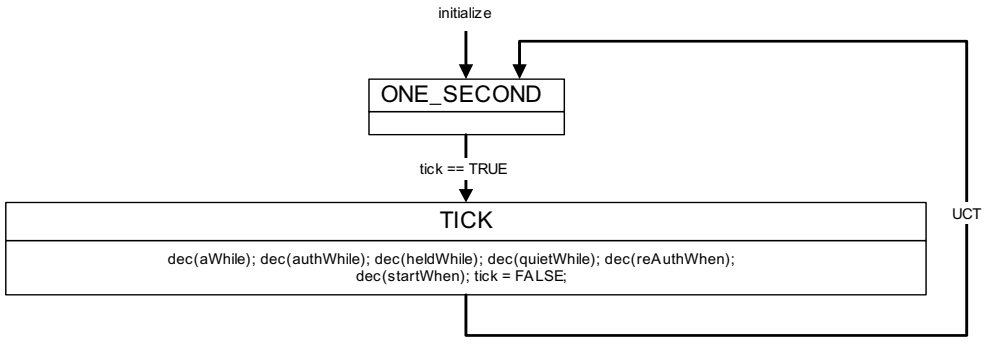


Figure 8-9—Port Timers state machine

NOTE—The state machine as defined includes timers used by state machines that are needed for both Authenticator and Supplicant functionality. Clearly, if only Authenticator functionality, or only Supplicant functionality, is implemented on a given Port, this state machine need only be concerned with the timers that are needed for the state machines involved. Similarly, if reauthentication is not supported, reAuthWhen timer is not required. Under these conditions, the timers needed for the unimplemented functionality can be considered to exist, but with their values permanently set to zero.

8.2.3.1 Variables used in the definition of the Port Timers state machine

8.2.3.1.1 Variables

— **tick.** This variable is set in response to a regular one-second tick generated by an external system clock function. Whenever the system clock generates a one-second tick, the tick variable is set TRUE. The variable is set FALSE by the operation of the state machine. The operation of the system clock function is not otherwise specified by this standard.

8.2.4 Authenticator PAE state machine

The Authenticator PAE state machine has the following states:

a) INITIALIZE (see 8.2.4.3)

b) DISCONNECTED (see 8.2.4.4)

c) RESTART (see 8.2.4.5)

d) CONNECTING (see 8.2.4.6)

e) AUTHENTICATING (see 8.2.4.7)

f) AUTHENTICATED (see 8.2.4.8)

g) ABORTING (see 8.2.4.9)

h) HELD (see 8.2.4.10)

i) FORCE_AUTH (see 8.2.4.11)

j) FORCE_UNAUTH (see 8.2.4.12)

The Authenticator PAE state machine shall implement the function specified by the state diagram contained in Figure 8-10 and the attendant definitions contained in 8.2.2 and 8.2.4.1.

8.2.4.1 Variables, constants, and procedures used in the definition of the Authenticator PAE state machine

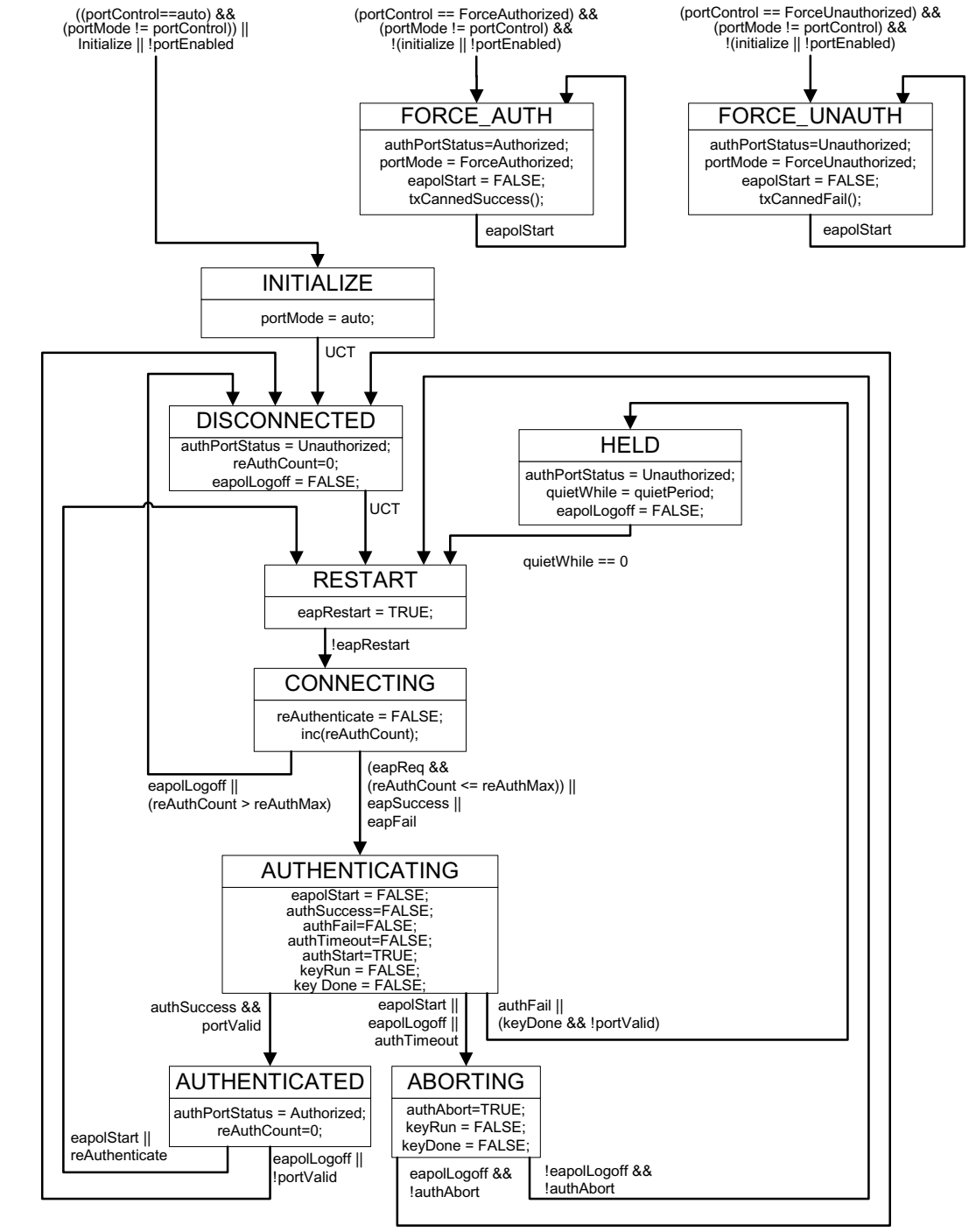
8.2.4.1.1 Variables

a) **eapolLogoff.** This variable is set TRUE if an EAPOL PDU carrying a Packet Type of EAPOL-Logoff is received. It is set FALSE by the operation of the Authenticator PAE state machine.

b) **eapolStart.** This variable is set TRUE if an EAPOL PDU carrying a Packet Type of EAPOL-Start is received. It is set FALSE by the operation of the Authenticator PAE state machine.

c) **eapReq.** This variable is set TRUE by the higher layer when it has an EAP frame to be sent to the Supplicant. It is set to FALSE by the Backend Authentication state machine when the EAP-frame has been transmitted.

d) **eapRestart.** This variable is set to TRUE by the Authenticator state machine to signal it is restarting its state machine due to an EAPOL packet, a timeout, or an initialization event.



The following abbreviation is used in this diagram:
inc(x): {x = x + 1; If (X > 255) then (x = 0);}

Figure 8-10—Authenticator PAE state machine

e) **portMode.** Used in conjunction with authPortControl to switch between the Auto and non-Auto modes of operation of the Authenticator PAE state machine. This variable can take the following values:

1) **ForceUnauthorized.** The controlled Port is required to be held in the Unauthorized state.

2) **ForceAuthorized.** The controlled Port is required to be held in the Authorized state.

3) **Auto.** The controlled Port is set to the Authorized or Unauthorized state in accordance with the outcome of an authentication exchange between the Supplicant and the Authentication Server.

f) **reAuthCount.** This variable counts the number of times the CONNECTING state is re-entered. If the count exceeds reAuthMax, it forces the Port to become Unauthorized before further attempts to authenticate can be made.

8.2.4.1.2 Constants

a) **quietPeriod.** The initialization value used for the quietWhile timer. Its default value is 60 s; it can be set by management to any value in the range from 0 to 65 535 s.

NOTE 1— The Authenticator may increase the value of quietPeriod per Port to ignore authorization failures for longer periods of time after a number of authorization failures have occurred.

b) **reAuthMax.** The number of reauthentication attempts that are permitted before the Port becomes Unauthorized. The default value of this constant is 2.

NOTE 2—These are constants in the sense that the state machines do not modify their values; however, their values may be modified by management.

8.2.4.1.3 Procedures

a) **txCannedFail**. An EAPOL frame of type EAP-Packet, containing an EAP Failure packet constructed by the Authenticator, is transmitted to the Supplicant. In the case that no EAP communication was taking place on the port, then any value of Id may be used in the identifier field of the EAP frame. In the case that there was an EAP communication taking place on the port, then the value of the Identifier field in the EAP packet is set to a value that is different from the last delivered EAPOL frame of type EAP-Packet.

b) **txCannedSuccess**. An EAPOL frame of type EAP-Packet, containing an EAP Success packet constructed by the Authenticator, is transmitted to the Supplicant. In the case that no EAP communication was taking place on the port, then any value of ID may be used in the identifier field of the EAP frame. In the case that there was an EAP communication taking place on the port, then the value of the Identifier field in the EAP packet is set to a value that is different from the last delivered EAPOL frame of type EAP-Packet.

8.2.4.2 Counters maintained by the Authenticator PAE state machine

The following counters may be maintained by the Authenticator PAE state machine for diagnostic purposes. The values of these counters may be made available to management via the management operations specified in Clause 9. The count values are assumed to roll over to zero when a counter is incremented beyond the maximum value specified for the associated managed object.

NOTE—For clarity, the actions involved in incrementing these counters are not shown on the accompanying state machines, as in some cases, inclusion of the necessary actions would necessitate the introduction of additional states in the diagrams.

8.2.4.2.1 authEntersConnecting

Counts the number of times that the state machine transitions to the CONNECTING state from any other state.

8.2.4.2.2 authEapLogoffsWhileConnecting

Counts the number of times that the state machine transitions from CONNECTING to DISCONNECTED as a result of receiving an EAPOL-Logoff message.

8.2.4.2.3 authEntersAuthenticating

Counts the number of times that the state machine transitions from CONNECTING to AUTHENTICATING, as a result of an EAP-Response/Identity message being received from the Supplicant.

8.2.4.2.4 authAuthSuccessesWhileAuthenticating

Counts the number of times that the state machine transitions from AUTHENTICATING to AUTHENTICATED, as a result of the Backend Authentication state machine indicating successful authentication of the Supplicant (authSuccess = TRUE).

8.2.4.2.5 authAuthTimeoutsWhileAuthenticating

Counts the number of times that the state machine transitions from AUTHENTICATING to ABORTING, as a result of the Backend Authentication state machine indicating authentication timeout (authTimeout = TRUE).

8.2.4.2.6 authAuthFailWhileAuthenticating

Counts the number of times that the state machine transitions from AUTHENTICATING to HELD, as a result of the Backend Authentication state machine indicating authentication failure (authFail = TRUE).

8.2.4.2.7 authAuthEapStartsWhileAuthenticating

Counts the number of times that the state machine transitions from AUTHENTICATING to ABORTING, as a result of an EAPOL-Start message being received from the Supplicant.

8.2.4.2.8 authAuthEapLogoffWhileAuthenticating

Counts the number of times that the state machine transitions from AUTHENTICATING to ABORTING, as a result of an EAPOL-Logoff message being received from the Supplicant.

8.2.4.2.9 authAuthReauthsWhileAuthenticated

Counts the number of times that the state machine transitions from AUTHENTICATED to RESTART, as a result of a reauthentication request (reAuthenticate = TRUE).

8.2.4.2.10 authAuthEapStartsWhileAuthenticated

Counts the number of times that the state machine transitions from AUTHENTICATED to CONNECTING, as a result of an EAPOL-Start message being received from the Supplicant.

8.2.4.2.11 authAuthEapLogoffWhileAuthenticated

Counts the number of times that the state machine transitions from AUTHENTICATED to DISCONNECTED, as a result of an EAPOL-Logoff message being received from the Supplicant.

8.2.4.3 INITIALIZE

This state is entered from any other state if the portControl variable is set to Auto and the portMode variable is set to some other value, or the Port's MAC is inoperable, or the state machine is being initialized. The value of the portMode variable is set to Auto.

An unconditional transition to the DISCONNECTED state occurs when the initialization is complete and the MAC service associated with the Port is operable, unless portMode and portControl differ in value; in which case, a global transition to the FORCE_AUTH or FORCE_UNAUTH states will occur.

8.2.4.4 DISCONNECTED

This state is entered from the CONNECTING state, the AUTHENTICATED state, and the ABORTING state if an explicit logoff request is received from the Supplicant.

If, at any time, the port's MAC becomes inoperable (goes "down"), then the PAE will transition to this state.

The authPortStatus variable is set to Unauthorized in this state, thereby setting the value of AuthControlledPortStatus (see 6.4) to Unauthorized, the eapolLogoff variable is cleared and the reAuthCount is reset.

The state machine unconditionally transitions to the RESTART state.

8.2.4.5 RESTART

The RESTART state is entered when the Authenticator PAE needs to inform the higher layer that it has restarted. This happens unconditionally from the DISCONNECTED state, and as a result of receiving an eapolStart or reAuthenticate signal while in the AUTHENTICATE state, and after leaving the ABORTING state because of anything other than an eapolLogoff from the Supplicant. This state is also entered from the HELD state on expiry of the quietWhile timer.

The eapRestart variable is set TRUE to signal to the higher layer that the Authenticator PAE has restarted. This state will exit to CONNECTING when EAP has acknowledged the restart by resetting eapRestart to FALSE.

8.2.4.6 CONNECTING

In this state, the Port is operable, the higher layer is in sync and ready to attempt to establish communication with a Supplicant.

If an EAPOL-Logoff frame is received, the state machine transitions to DISCONNECTED, in order to force the Port to the Unauthorized state.

NOTE—As this state can be entered from RESTART via AUTHENTICATED on a reauthentication request or an EAPOL-Start message, the Port can be in the Authorized state while this state machine is CONNECTING. This allows repeated reauthentication to take place without forcing the Port to be Unauthorized.

If the higher layer is ready to send an initial EAP-Request message, or it has decided to complete EAP with either a Success or Fail, the state machine transitions to the AUTHENTICATING state.

EAP is responsible for retransmitting the initial and subsequent EAP-Request messages. However, the state keeps a tally of the number of times that it has been re-entered without forcing the authPortStatus to Unauthorized; if this tally (reAuthCount) exceeds a maximum value (reAuthMax), then a transition to DISCONNECTED occurs in order to force the authPortStatus to Unauthorized and to clear the reAuthCount.

8.2.4.7 AUTHENTICATING

In this state, a Supplicant is being authenticated. The variables authSuccess, authTimeout, authFail, keyRun, and keyDone are set to FALSE, and the variable authStart is set to TRUE in order to signal to the Backend Authentication state machine to start the authentication procedure. This procedure can generate one of the following results:

a) The authentication procedure terminates due to excessive timeouts in the sequence of requests and responses. The variable authTimeout is set TRUE, causing the state machine to transition to the ABORTING state.

b) The authentication procedure terminates due to the Authentication Server returning a Reject message to the Authenticator. The variable authFail is set TRUE, causing the state machine to transition to the HELD state.

c) The authentication procedure terminates due to the Authentication Server returning an Accept message to the Authenticator. The variable authSuccess is set TRUE, causing the state machine to transition to the AUTHENTICATED state if portValid is also TRUE.

If an EAPOL-Start or EAPOL-Logoff frame is received, then the state machine transitions to the ABORTING state in order to abort the authentication procedure. If a reauthentication request is received, it is not acted upon immediately, but it is remembered until the authentication process is completed.

8.2.4.8 AUTHENTICATED

In this state, the Authenticator has successfully authenticated the Supplicant and the portValid variable has become TRUE. The authPortStatus variable is set to the Authorized state and the reauth count is reset in order to allow subsequent reauthentication requests.

If a reauthentication request or an EAPOL-Start frame is received, the state machine transitions to the RESTART state in order to reauthenticate the Supplicant. If an EAPOL-Logoff frame is received, or if portValid becomes FALSE, the state machine transitions to the DISCONNECTED state in order to force the authPortStatus to Unauthorized before attempting to reauthenticate the Supplicant.

8.2.4.9 ABORTING

In this state, the authentication procedure is being prematurely aborted due to receipt of an EAPOL-Start frame, an EAPOL-Logoff frame, or an authTimeout. The authAbort variable is set TRUE to signal to the Backend Authentication state machine that it should terminate the authentication procedure.

Exit from this state to the RESTART state occurs once the authentication procedure has aborted. Exit from this state to the DISCONNECTED state occurs if an EAPOL-Logoff caused entry to this state, or if one is received while in this state; this ensures that the Port state is forced to Unauthorized in this case.

8.2.4.10 HELD

In this state, the state machine ignores and discards all EAPOL packets, so as to discourage brute force attacks. This state is entered from the AUTHENTICATING state following an authentication failure.

The authPortStatus variable is set to Unauthorized, and the quietWhile timer is started using the value quietPeriod. At the expiration of the quietWhile timer, the state machine transitions to the RESTART state.

8.2.4.11 FORCE_AUTH

This state is entered from any other state if the following four conditions are all TRUE:

a) The portControl variable is set to ForceAuthorized.

b) The portMode variable is set to some value other than ForceAuthorized.

c) The Port's MAC is operable.

d) The state machine is not being initialized.

The authPortStatus is set to Authorized, and a "canned" EAP Success packet (i.e., a message constructed by the Authenticator rather than sent by the Authentication Server) is sent to the Supplicant. If an EAPOL-Start message is received from the Supplicant, the state is re-entered and a further EAP Success message is sent.

The effect of this set of actions is to force the Port state to Authorized, and to reflect this state back to the Supplicant if it should initiate authentication, thereby removing unnecessary delays before the Supplicant assumes that it has been authenticated successfully.

8.2.4.12 FORCE_UNAUTH

This state is entered from any other state if the following four conditions are all TRUE:

a) The portControl variable is set to ForceUnauthorized.

b) The portMode variable is set to some value other than ForceUnauthorized.

c) The Port's MAC is operable.

d) The state machine is not being initialized.

The authPortStatus is set to Unauthorized, and a "canned" EAP Failure packet (i.e., a message constructed by the Authenticator rather than sent by the Authentication Server) is sent to the Supplicant. If EAP-Start messages are received from the Supplicant, the state is re-entered and further EAP Failure messages are sent.

The effect of this set of actions is to force the Port state to Unauthorized, and to reflect this state back to the Supplicant if it should initiate authentication, thereby removing unnecessary delays before the Supplicant discovers that it has not been authenticated successfully.

8.2.5 Authenticatior Key Transmit state machine

The Authenticator Key Transmit state machine shall implement the function specified by the state diagram contained in Figure 8-11 and the attendant definitions contained in 8.2.2 and 8.2.5.1. Alternative Authenticator key transmit machines defined by IEEE Std 802.11 may be used in place of the machine defined in this clause. Any alternative transmit key machine must adhere to the interface defined in 8.1.9.

NOTE 1—When used with IEEE Std 802.11, the Authenticator Key Transmit state machine can be used to transmit group keys as well as unicast keys.

The Authenticator Key Transmit state machine transmits EAPOL-Key PDUs (see 7.5.4) to the Supplicant, if the following conditions are all true:

a) The Port is not undergoing initialization.

b) The portControl setting is Auto.

c) Key transmission has been enabled.

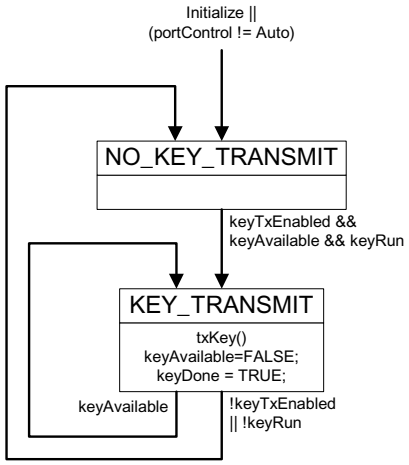


Figure 8-11—Authenticator Key Transmit state machine

d) There is new key material available for transmission.

e) The Backend Authentication state machine has asserted keyRun to indicate that the key machine may run.

NOTE 2—As described in IETF RFC 3579, in addition to containing EAP-Message attributes, RADIUS messages may also contain other attributes, including the information necessary to generate an EAPOL-Key frame. Proper processing of RADIUS messages requires the Authenticator to process EAP-Message attributes first. The function of the keyRun variable ensures the EAP-Success message will be sent prior to the sending of EAPOL-Key messages.

8.2.5.1 Variables, constants, and procedures used in the definition of the Authenticator Key Transmit state machine

8.2.5.1.1 Procedures

— **txKey**(). An EAPOL frame of type EAPOL-KEY, containing an EAPOL-Key packet, is transmitted to the Supplicant.

8.2.6 Supplicant Key Transmit state machine

The Supplicant Key Transmit state machine shall implement the function specified by the state diagram contained in Figure 8-12 and the attendant definitions contained in 8.2.2 and 8.2.6.1. Alternative Supplicant Key Transmit machines defined by IEEE Std 802.11 may be used in place of the machine defined in this clause. Any alternative transmit key machine must adhere to the interface defined in 8.1.9.

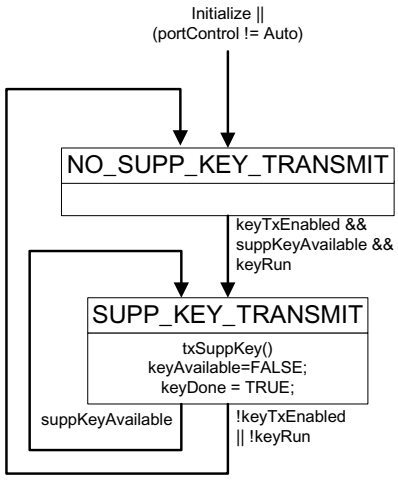


Figure 8-12—Supplicant Key Transmit state machine

The Supplicant Key Transmit state machine transmits EAPOL-Key PDUs (see 7.5.4) to the Authenticator, if the following conditions are all true:

a) The Port is not undergoing initialization.

b) Key transmission has been enabled.

c) There is new key material available for transmission.

d) The Supplicant machines have asserted keyRun to indicate that the key machine may run.

8.2.6.1 Variables, constants, and procedures used in the definition of the Supplicant Key Transmit state machine

8.2.6.1.1 Variables

— **suppKeyAvailable.** Set to TRUE by an external entity when there is a new session key available that can be used by the key machine to begin a new key exchange. Set to FALSE when the Authenticator Key Transmit state machine has transmitted the key value.

8.2.6.1.2 Constants

— **keyTxEnabled.** Reflects the current value of the KeyTransmissionEnabled parameter (see 8.1.9).

NOTE—These are constants in the sense that the state machines do not modify their values; however, their values may be modified by management.

8.2.6.1.3 Procedures

— **txSuppKey().** An EAPOL frame of type EAPOL-KEY, containing an EAPOL-Key packet, is transmitted to the Authenticator.

8.2.7 Key Receive state machine

The Key Receive state machine shall implement the function specified by the state diagram contained in Figure 8-13 and the attendant definitions contained in 8.2.2 and 8.2.7.1. Alternative Key Receive machines defined by IEEE Std 802.11 may be used in place of the machine defined in this clause. Any alternative Key Receive machine must adhere to the interface defined in 8.1.9.

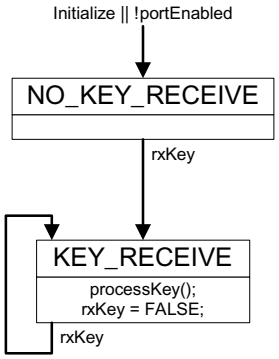


Figure 8-13—Key Receive state machine

The Key Receive state machine allows EAPOL-Key PDUs (see 7.5.4) to be received from the Supplicant or Authenticator and processed in accordance with any encryption mechanisms being employed by the Authenticator or Supplicant.

8.2.7.1 Variables and procedures used in the definition of the Key Receive state machine

8.2.7.1.1 Variables

— **rxKey.** This variable is set TRUE if an EAPOL-Key message is received by the Supplicant or Authenticator (see 8.1.9). It is set FALSE when the Key Receive state machine has transmitted the key value.

8.2.7.1.2 Procedures

— **processKey().** This procedure causes the Supplicant or Authenticator to act on the key information provided in a received EAPOL-Key message. If the Supplicant or Authenticator has no use for the key information provided, the EAPOL_KEY message is discarded.

NOTE—The use of such key information by the system is dependent on the encryption mechanism(s) that is (are) being employed by the system, if any.

8.2.8 Reauthentication Timer state machine

The Reauthentication Timer state machine for a given Port is responsible for ensuring that periodic reauthentication of the Supplicant takes place, if periodic reauthentication is enabled (reAuthEnabled is TRUE). The state machine is held in the INITIALIZE state until such a time as the portControl for the Port is Auto, the portStatus for the Port becomes Authorized, the port is not being initialized, and the reAuthEnabled control is TRUE. The reAuthWhen timer is set to its initial value; when it expires, the state machine will then transition to the REAUTHENTICATE state, setting the reAuthenticate variable TRUE, and then transitioning back to INITIALIZE for a further timer cycle.

The Reauthentication Timer state machine shall implement the function specified by the state diagram contained in Figure 8-14 and the attendant definitions contained in 8.2.2 and 8.2.8.1.

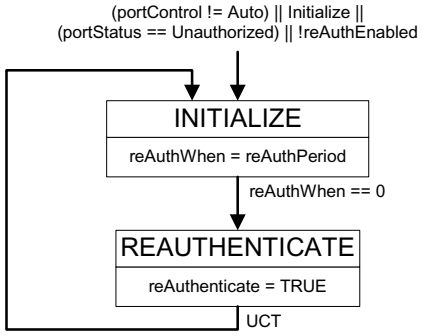


Figure 8-14—Reauthentication Timer state machine

8.2.8.1 Constants used in the definition of the Reauthentication Timer state machine

a) **reAuthPeriod.** A constant that defines a nonzero number of seconds between periodic reauthentication of the Supplicant. The default value is 3600 s.

b) **reAuthEnabled.** A constant that defines whether regular reauthentication will take place on this Port. A value of TRUE enables reauthentication; FALSE disables reauthentication.

NOTE—These are constants in the sense that the state machines do not modify their values; however, their values may be modified by management.

8.2.9 Backend Authentication state machine

The Backend Authentication state machine has the following states:

a) REQUEST (see 8.2.9.3)

b) RESPONSE (see 8.2.9.4)

c) SUCCESS (see 8.2.9.5)

d) FAIL (see 8.2.9.6)

e) TIMEOUT (see 8.2.9.7)

f) IDLE (see 8.2.9.8)

g) INITIALIZE (see 8.2.9.9)

h) IGNORE (see 8.2.9.10)

The Backend Authentication state machine shall implement the function specified by the state diagram contained in Figure 8-18 and the attendant definitions contained in 8.2.2 and 8.2.9.1.

8.2.9.1 Variables, constants, and procedures used in the definition of the Backend Authentication state machine

8.2.9.1.1 Variables

a) **eapNoReq.** This variable is set TRUE by the higher layer when it has no EAP frame to be sent to the Supplicant in response to the last EAP frame sent by the Supplicant.

b) **eapReq.** This variable is set TRUE by the higher layer when it has an EAP frame to be sent to the Supplicant.

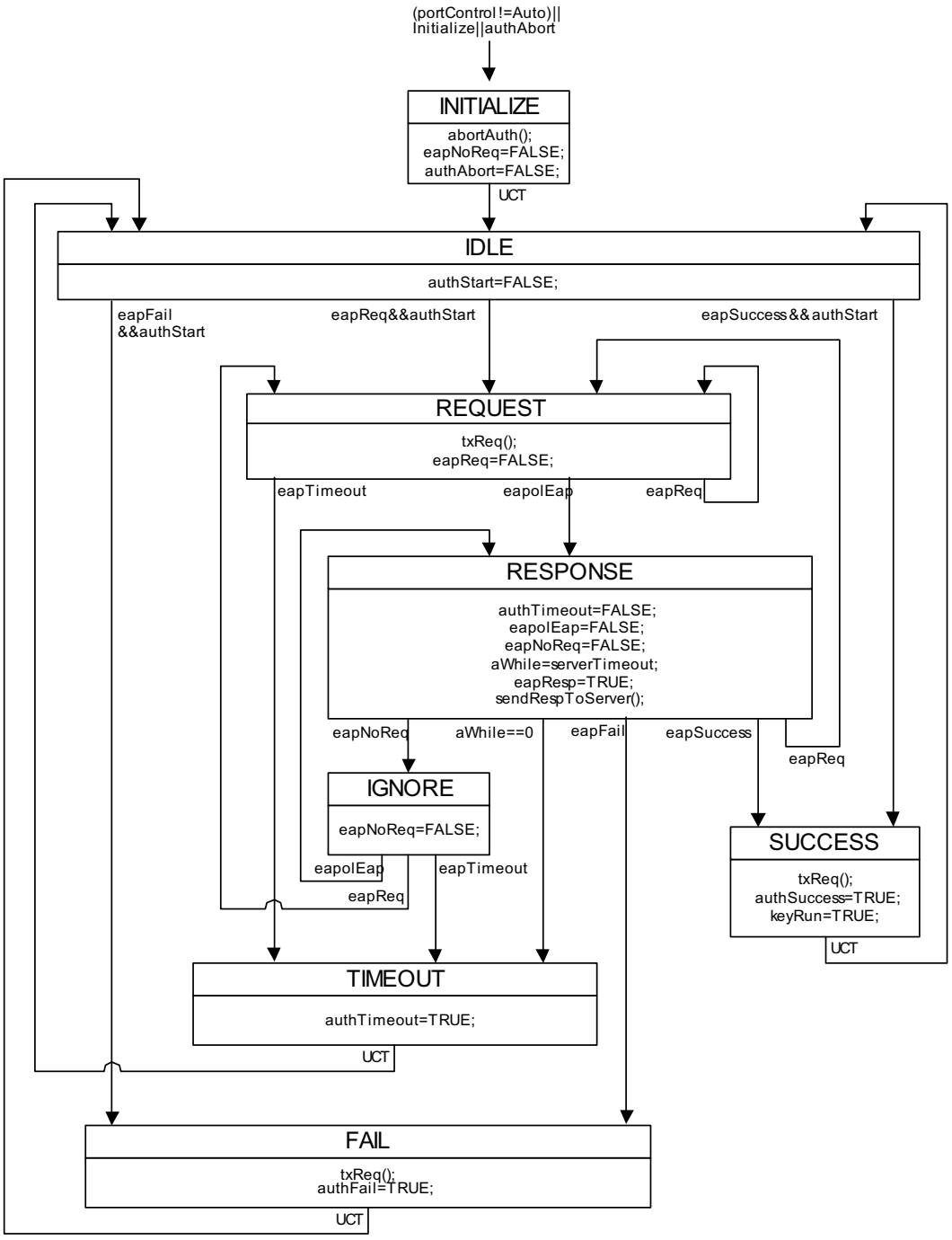


Figure 8-15—Backend Authentication state machine

c) **eapResp.** Set TRUE by the Backend Authenticator state machine to indicate there is a new EAP frame available for the higher layer to process. Set to FALSE by EAP to indicate it has seen the EAP frame.

8.2.9.1.2 Constants

a) **serverTimeout.** The initialization value used for the aWhile timer when timing out the higher layer. This timer should be longer than the longest time that the higher layer would take to exhaust all of its retries to the Authentication Server with its current timeout period. The aWhile timer has a default value of 30 s; however, the timeout value may be adjusted to take into account EAP/AAA settings. It can be set by management to any value in the range from 1 to X s, where X is an implementation-dependent value.

NOTE—These are constants in the sense that the state machines do not modify their values; however, their values may be modified by management.

8.2.9.1.3 Procedures

a) **txReq(x).** An EAPOL frame of type EAP-Packet is transmitted to the Supplicant if one is available from the higher layer. EAP is not required to make an EAP-Packet available, in which case no EAPOL frame will be delivered.

b) **sendRespToServer().** The EAP frame most recently received from the Supplicant is delivered to EAP for processing.

c) **abortAuth.** This procedure allows the Backend Authentication state machine to release any system resources that may have been occupied during the current authentication session, prior to signaling to the Authentication state machine that the session has been successfully aborted.

8.2.9.2 Counters maintained by the Backend Authentication state machine

The following counters may be maintained by the Backend Authentication state machine for diagnostic purposes. The values of these counters may be made available to management via the management operations specified in Clause 9. The count values are assumed to roll over to zero when a counter is incremented beyond the maximum value specified for the associated managed object.

NOTE—For clarity, the actions involved in incrementing these counters are not shown on the accompanying state machines, as in some cases, inclusion of the necessary actions would necessitate the introduction of additional states in the diagrams.

8.2.9.2.1 backendResponses

Counts the number of times that the state machine sends a Supplicant's first response packet to the AAA client (i.e., executes sendRespToServer on entry to the RESPONSE state). Indicates that the Authenticator attempted communication with the Authentication Server via the AAA client.

8.2.9.2.2 backendAccessChallenges

Counts the number of times that the state machine receives the first request from the AAA client following the first response from the Supplicant (i.e., eapReq becomes TRUE, causing exit from the RESPONSE state). Indicates that the Authentication Server has communication with the Authenticator via the AAA client.

8.2.9.2.3 backendOtherRequestsToSupplicant

Counts the number of times that the state machine sends an EAP-Request packet following the first to the Supplicant (i.e., executes txReq on entry to the REQUEST state). Indicates that the Authentication Server chose an EAP-method.

8.2.9.2.4 backendAuthSuccesses

Counts the number of times that the state machine receives a success indication from the AAA client (i.e., eapSuccess becomes TRUE, causing a transition from RESPONSE to SUCCESS). Indicates that the Supplicant has successfully authenticated to the Authentication Server.

8.2.9.2.5 backendAuthFails

Counts the number of times that the state machine receives a failure message from the AAA client (i.e., eapFail becomes TRUE, causing a transition from RESPONSE to FAIL). Indicates that the Supplicant has not authenticated to the Authentication Server.

8.2.9.3 REQUEST

In this state, the state machine has received an EAP Request packet from the higher layer and is relaying that packet to the Supplicant as an EAPOL-encapsulated frame. In the event that the EAP Request packet is lost, EAP (which handles retransmission) will signal a re-send with the same EAP Request packet. The state machine transitions back into the REQUEST state.

If EAP has reached its maximum number of retransmissions it will signal a timeout, causing the state machine to transition to the TIMEOUT state.

If an EAPOL frame is received from the Supplicant, containing an EAP Response packet, the state machine transitions to the RESPONSE state.

8.2.9.4 RESPONSE

In this state, the state machine has received an EAPOL-encapsulated EAP Response packet from the Supplicant, and is relaying the EAP packet to the higher layer to be relayed on to the Authentication Server, and is awaiting instruction from the higher layer as to what to do next. The variable aWhile is used to time out the response from the higher layer.

NOTE—The state machine as described assumes that the communication path between the Authenticator and the Authentication Server via the AAA protocol is inherently reliable; i.e., the state machine does not define any retransmission regime for this aspect of the communication path; this is assumed to be the responsibility of the AAA protocol. The value of serverTimeout should therefore be long enough to cover the duration of this transmission regime. If the AAA protocol is not providing a reliable communication path, then the implementation may adopt a retransmission or failover strategy that is suitable for the type of network connection in use, as described in IETF RFC 3579. Where such a retransmission strategy is adopted, the point at which the state machine determines that a server timeout has occurred will depend on the strategy adopted, rather than a simple timing out of the aWhile timer.

If a timeout occurs, the state machine transitions to the TIMEOUT state.

If the higher layer indicates that authentication has been successful (eapSuccess) then the state machine transitions to the SUCCESS state.

If the higher layer indicates that the authentication has been unsuccessful (eapFail) then the state machine transitions to the FAIL state.

If the higher layer decides to ignore the previous EAP Response message received, it will do so by asserting eapNoReq and the state machine will transition to the IGNORE state.

If a further EAP Request packet is ready for transmission from the higher layer, the state machine transitions to the REQUEST state in order to relay the request to the Supplicant.

8.2.9.5 SUCCESS

The state machine sets the global variable authSuccess TRUE in order to signal to the Authenticator state machine that the authentication session has terminated successfully, and it transmits the final EAP message from the AAA client that was encapsulated in the Accept message from the Authentication Server. Typically, this is an EAP Success packet.

The keyRun variable is set TRUE in order to assure the sequence of sending the final EAP message prior to initiating the Authenticator Key Transmit machines. The state machine then transitions to the IDLE state.

NOTE—An accept can be received from the Authentication Server with or without an encapsulated EAP packet, and if there is an EAP packet present, it may or may not be an EAP success packet. As noted in IETF RFC 3579, the responsibility for avoiding conflicts lies with the Authentication Server.

8.2.9.6 FAIL

The state machine sets the global variable authFail TRUE in order to signal to the Authenticator state machine that the authentication session has terminated with an authentication failure, and it transmits the final EAP message from the AAA client that was encapsulated in the Reject message from the Authentication Server. Typically, this is an EAP Failure packet.

The state machine then transitions to the IDLE state.

NOTE—A reject can be received from the Authentication Server with or without an encapsulated EAP packet, and if there is an EAP packet present, it may or may not be an EAP success packet. As noted in IETF RFC 3579, the responsibility for avoiding conflicts lies with the Authentication Server.

8.2.9.7 TIMEOUT

This state may be reached either due to an IEEE 802.1X timeout or due to a timeout signaled by the higher layer via the eapTimeout signal. The state machine sets the global variable authTimeout TRUE in order to signal to the Authenticator state machine that the authentication session has terminated with a timeout. The state machine then transitions to the IDLE state.

8.2.9.8 IDLE

In this state, the state machine is waiting for the Authenticator state machine to signal the start of a new authentication session. When authStart becomes TRUE, indicating that the higher layer is ready to initiate a session, the state machine transitions to the REQUEST state in order to relay the initial EAP packet to the Supplicant. The higher layer may decide to bypass the authentication session entirely by asserting either eapSuccess or eapFail immediately and causing transitions to SUCCESS or FAIL respectively.

8.2.9.9 INITIALIZE

This state is entered from any other state if a system initialization occurs, or if the Authenticator state machine sets the global variable authAbort TRUE, signaling that the current authentication session is to be terminated. The abortAuth procedure is used to release any system resources that may have been occupied by the session.

The state machine transitions to IDLE once the variables initialize, eapNoReq, and authAbort are all FALSE.

NOTE—The nature of the abortAuth procedure is implementation dependent.

8.2.9.10 IGNORE

This state is entered when the higher layer has decided to ignore the previous EAP Response message received from the Supplicant. In this case, the higher layer indicates that no request is being sent to follow the previous EAP Response message by asserting eapNoReq. This state clears the eapNoReq variable and transitions back to the RESPONSE state when the next EAP Response message is received.

NOTE—It is rare for EAP to ignore an EAP Response packet from the Supplicant. Potential causes of such an event include denial of service attacks from rogue Supplicants or malformed EAP Response messages.

8.2.10 Controlled Directions state machine

The Controlled Directions state machine for a given Port is responsible for ensuring that the value of the OperControlledDirections parameter for the Port correctly reflects the current state of the AdminControlledDirections parameter coupled with the operational state of the MAC and the presence or absence of a Bridge (see 6.5).

If OperControlledDirections is set to IN on a Bridge Port, this allows the Bridge to forward frames received from its other Bridge Ports onto that Port, but prevents frames received on that Port (including BPDUs) from being processed or forwarded by the Bridge. In order to prevent the possibility of configuring inadvertent loops as a result of connecting a Bridge to a Bridge Port that is set to IN, OperControlledDirections is forced to BOTH if the operEdge variable (see Clause 17 of IEEE Std 802.1D) for the Port is FALSE.

The Controlled Directions state machine shall implement the function specified by the state diagram contained in Figure 8-16 and the attendant definitions contained in 8.2.2 and 8.2.10.1.

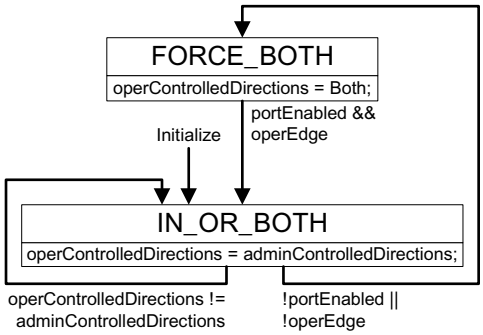


Figure 8-16—Controlled Directions state machine

If this state machine is implemented on a Bridge Port, and the Port supports the use of AdminControlledDirections and OperControlledDirections parameter values of In and Both (see 6.5), then the Bridge Detection state machine shall also be implemented on the Uncontrolled Port, in accordance with the requirements stated in Clause 17 of IEEE Std 802.1D.

The IN_OR_BOTH state is entered from any state on initialization of the state machine. The value of operControlledDirections is set to the value of adminControlledDirections on entry to this state. Exit from this state to FORCE_BOTH occurs if the Port's MAC becomes inoperable, or if the Port is a Bridge Port and is not an Edge Port. Re-entry to this state occurs if the values of the adminControlledDirections and oper-ControlledDirections parameters differ.

On entry to the FORCE_BOTH state, the operControlledDirections parameter is set to BOTH. Exit from FORCE_BOTH to IN_OR_BOTH occurs if the entry conditions to FORCE_BOTH are no longer present.

8.2.10.1 Variables used in the definition of the Controlled Directions state machine

a) **adminControlledDirections.** The value of the AdminControlledDirections parameter (see 6.5); it can take the values Both or In. This parameter is used, but not modified, by this state machine; its value may be changed only by management means.

b) **operControlledDirections.** The value of the OperControlledDirections parameter (see 6.5); it can take the values Both or In. The value of this parameter is determined by the operation of the state machine.

c) **operEdge.** The value of the operEdge variable maintained by a Bridge Port (see Clause 17 of IEEE Std 802.1D). If this Port is not a Bridge Port, then the value of this variable is TRUE.

8.2.11 Supplicant PAE state machine

The Supplicant PAE state machine has the following states:

a) LOGOFF (see 8.2.11.2)

b) DISCONNECTED (see 8.2.11.3)

c) CONNECTING (see 8.2.11.4)

d) AUTHENTICATING (see 8.2.11.5)

e) HELD (see 8.2.11.6)

f) AUTHENTICATED (see 8.2.11.7)

g) RESTART (see 8.2.11.8)

h) S_FORCE_AUTH (see 8.2.11.9)

i) S_FORCE_UNAUTH (see 8.2.11.10)

The Supplicant PAE state machine shall implement the function specified by the state diagram contained in Figure 8-17 and the attendant definitions contained in 8.2.2 and 8.2.11.1.

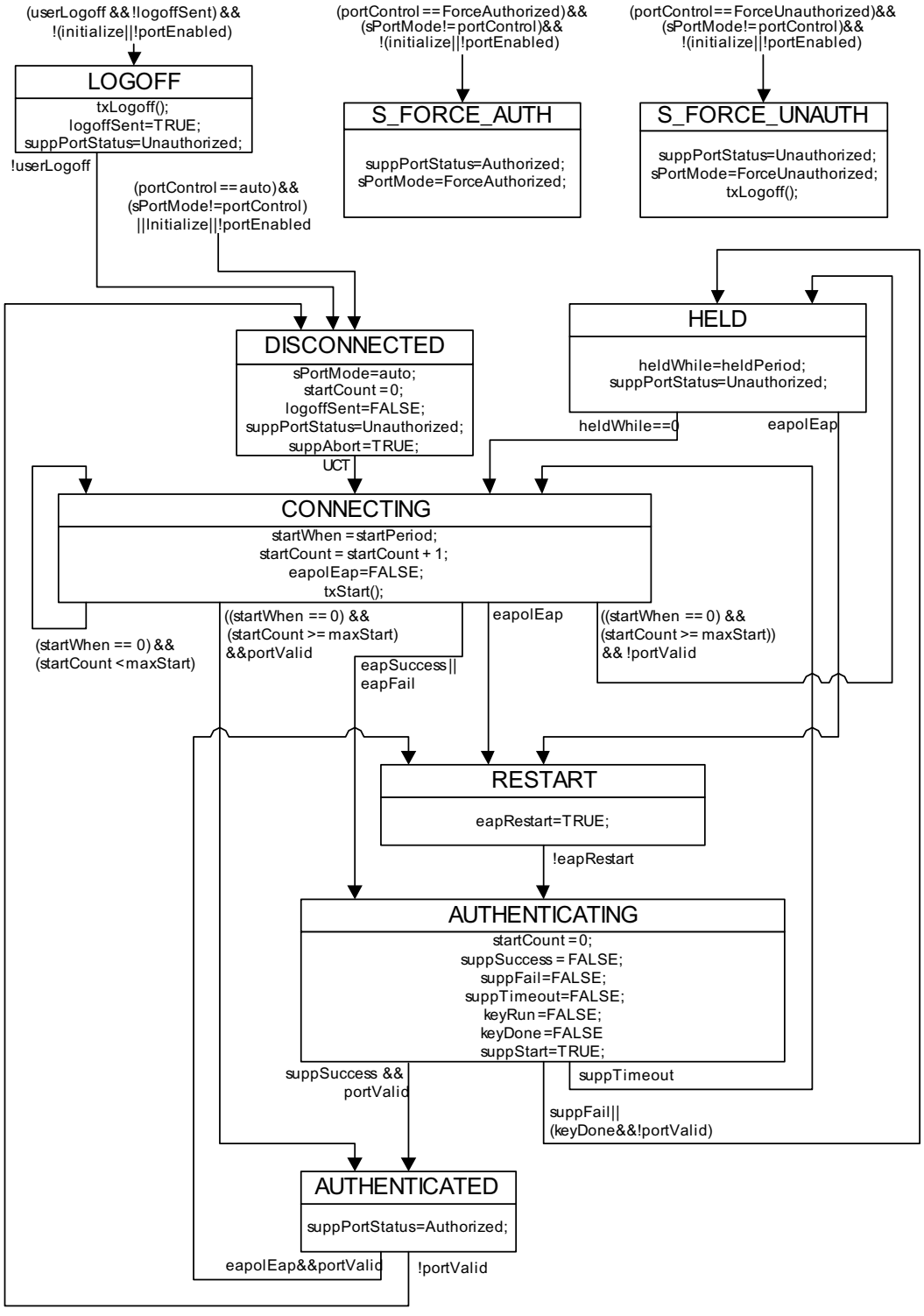


Figure 8-17—Supplicant PAE state machine

8.2.11.1 Variables, constants, and procedures used in the definition of the Supplicant PAE state machine

8.2.11.1.1 Variables

a) **eapRestart.** This variable is set to TRUE by the Supplicant state machine to signal it is restarting its state machine (due to an EAPOL packet or timeout) and has a new EAP Request for EAP to process. This variable is set FALSE by the higher layer when it is ready to establish an authentication session.

b) **logoffSent.** Indicates whether an EAPOL-Logoff message has been sent from within the LOGOFF state, thereby preventing repeated re-entry to the state and consequent multiple transmission of logoff messages.

c) **sPortMode.** Used in conjunction with authPortControl to switch between the Auto and non-Auto modes of operation of the Supplicant PAE state machine. This variable can take the following values:

   1) **ForceUnauthorized.** The controlled Port is required to be held in the Unauthorized state.

   2) **ForceAuthorized.** The controlled Port is required to be held in the Authorized state.

   3) **Auto.** The controlled Port is set to the Authorized or Unauthorized state in accordance with the outcome of an authentication exchange between the Supplicant and the Authentication Server.

d) **startCount.** This variable is used to count the number of EAPOL-Start messages that have been sent without receiving a response from the Authenticator.

e) **userLogoff.** This variable is controlled externally to the state machine and reflects the operation of the process in the Supplicant System that controls the logged on/logged off state of the user of the system. Its value is set FALSE if the Supplicant System considers that its user is logged on; its value is set TRUE if the Supplicant System considers that its user is logged off.

NOTE—The nature of the user of a system, and the process used for logging the user on/off, is system dependent and is outside the scope of this standard. For example, systems that do not have "human" users, such as Bridges, might consider the "user" to be permanently logged on; systems that have "human" users, such as workstations, might make use of the services provided by their operating system to perform user logon/logoff.

8.2.11.1.2 Constants

a) **heldPeriod.** The initialization value used for the heldWhile timer. Its default value is 60 s.

b) **startPeriod.** The initialization value used for the startWhen timer. Its default value is 30 s.

c) **maxStart.** The maximum number of successive EAPOL-Start messages that will be sent before the Supplicant assumes that there is no Authenticator present. Its default value is 3.

NOTE—These are constants in the sense that the state machines do not modify their values; however, their values may be modified by management.

8.2.11.1.3 Procedures

a) **txStart().** An EAPOL frame of type EAPOL-Start is transmitted to the Authenticator.

b) **txLogoff().** An EAPOL frame of type EAPOL-Logoff is transmitted to the Authenticator.

8.2.11.2 LOGOFF

This state is entered if the user of the System requests an explicit logoff. An EAPOL-Logoff packet is transmitted to the Authenticator. The DISCONNECTED state is entered when userLogoff becomes FALSE.

8.2.11.3 DISCONNECTED

This state is entered from any other state when the MAC service associated with the Port is inoperable, or when the System is initialized or reinitialized. It is also entered from LOGOFF after an explicit logoff request from the user of the System, and from the AUTHENTICATED state if portValid becomes FALSE.

When the initialization is complete and the MAC service associated with the Port becomes operable, the state machine transitions to the CONNECTING state.

8.2.11.4 CONNECTING

In this state, the Port has become operable and the Supplicant is attempting to acquire an Authenticator.

An EAPOL-Start packet is transmitted by the Supplicant, and the startWhen timer is started, to cause retransmission if no response is received from the Authenticator. If the startWhen timer expires, the transmission is repeated up to a maximum of maxStart transmissions. If no response is received after maxStart transmissions, the state machine assumes that it is attached to a System that is not EAPOL aware, and transitions to AUTHENTICATED state if portValid is TRUE.

If an EAP-Request frame is received, the Supplicant transitions to the RESTART state.

If the higher layer has decided it is satisfied with an eapSuccess or eapFail, the Supplicant transitions directly to the AUTHENTICATING state.

8.2.11.5 AUTHENTICATING

In this state, an EAP Request packet has been received from the Authenticator. The suppStart global variable is asserted, indicating to the Supplicant Backend state machine that the process of responding to the request(s) from the Authenticator should be started. Exit from this state occurs once the Supplicant Backend state machine has completed its work, signaled by suppSuccess, suppFail, or suppTimeout being set TRUE. If suppSuccess becomes TRUE, this causes a transition to the AUTHENTICATED state only if portValid is also TRUE. If suppFail or suppTimeout become TRUE, the state machine transitions to HELD or CONNECTING respectively.

8.2.11.6 HELD

This state is entered from the AUTHENTICATING state following an authentication failure, signaled by the suppFail variable being set TRUE by the Supplicant Backend state machine. This state is also entered from the CONNECTING state if no Authenticator is acquired and the portValid variable is FALSE.

The state provides a delay period before the Supplicant will attempt to acquire an Authenticator. The heldWhile timer is started using the value heldPeriod. At the expiration of the heldWhile timer, the state machine transitions to the CONNECTING state.

If a Request packet is received from the Authenticator while in this state, the state machine transitions to the RESTART state.

8.2.11.7 AUTHENTICATED

To enter this state, portValid must be TRUE, and either the Supplicant has been successfully authenticated by the Authenticator, or it has assumed that the Authenticator is not EAPOL aware. The state is entered from the AUTHENTICATING state when the Supplicant Backend state machine asserts suppSuccess, or from the CONNECTING state if attempts to establish a dialogue with the Authenticator have been timed out. On

receiving an EAP-Request frame while portValid is asserted, the Supplicant transitions to the RESTART state.

8.2.11.8 RESTART

The RESTART state is entered when the Supplicant PAE needs to inform the higher layer that it has restarted. This happens when an EAP packet is received while in the AUTHENTICATED state, the CONNECTING state, or the HELD state.

The eapRestart variable is set TRUE to signal to the higher layer that the Supplicant PAE has restarted. This state will exit to AUTHENTICATING state when the higher layer has acknowledged the restart by resetting eapRestart to FALSE.

8.2.11.9 S_FORCE_AUTH

This state is entered from any other state if the following four conditions are all TRUE:

a) The portControl variable is set to ForceAuthorized.

b) The sPortMode variable is set to some value other than ForceAuthorized.

c) The Port's MAC is operable.

d) The state machine is not being initialized.

The suppPortStatus is set to Authorized, and the sPortMode variable is set to ForceAuthorized.

The effect of these actions is to force the Port state to Authorized.

8.2.11.10 S_FORCE_UNAUTH

This state is entered from any other state if the following four conditions are all TRUE:

a) The portControl variable is set to ForceUnauthorized.

b) The sPortMode variable is set to some value other than ForceUnauthorized.

c) The Port's MAC is operable.

d) The state machine is not being initialized.

The suppPortStatus is set to Unauthorized, and an EAPOL Logoff packet is sent to the Authenticator. The sPortMode variable is set to ForceUnauthorized.

The effect of this set of actions is to force the Port state to Unauthorized, and to reflect this state back to the Authenticator by issuing a logoff request.

8.2.12 Supplicant Backend state machine

The Supplicant Backend state machine has the following states:

a) REQUEST (see 8.2.12.2)

b) RESPONSE (see 8.2.12.3)

c) SUCCESS (see 8.2.12.4)

d) FAIL (see 8.2.12.5)

e) TIMEOUT (see 8.2.12.6)

f) IDLE (see 8.2.12.7)

g) INITIALIZE (see 8.2.12.8)

h) RECEIVE (see 8.2.12.9)

The Supplicant Backend state machine shall implement the function specified by the state diagram contained in Figure 8-18 and the attendant definitions contained in 8.2.2 and 8.2.12.1.

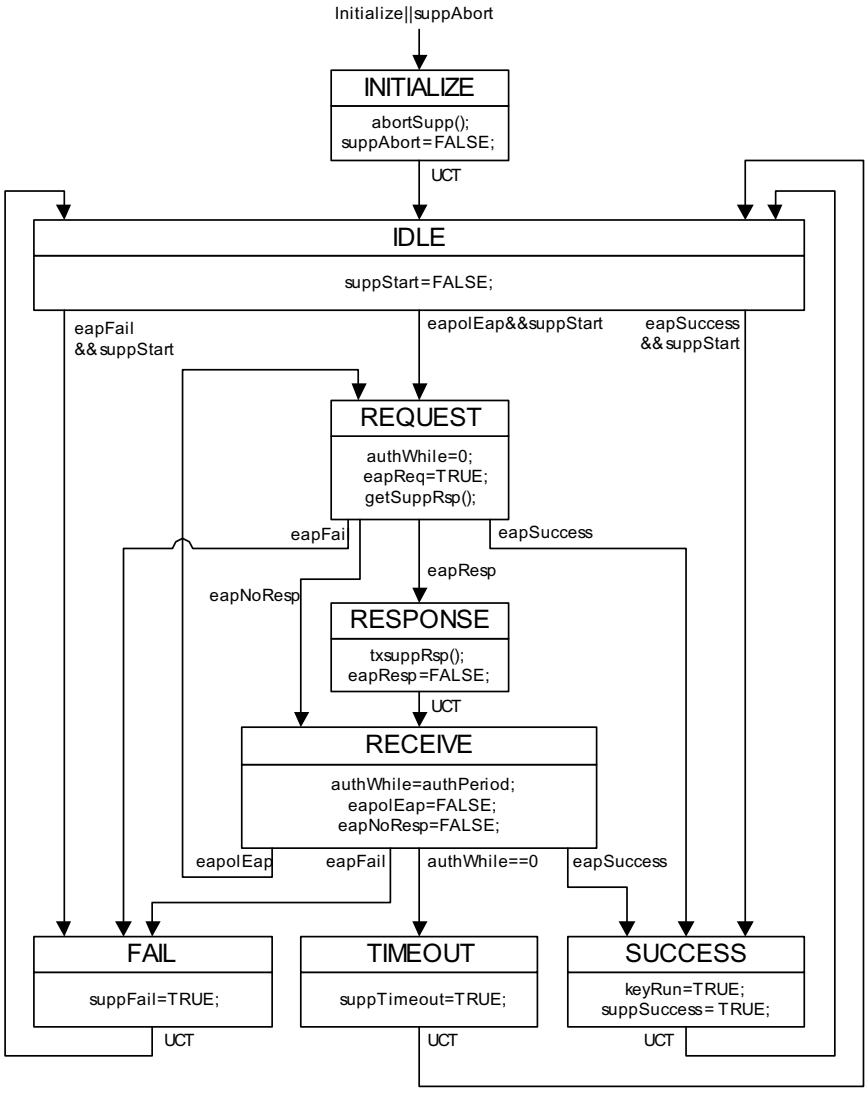


Figure 8-18—Supplicant Backend state machine

8.2.12.1 Variables, constants, and procedures used in the definition of the Supplicant Backend state machine

8.2.12.1.1 Variables

a) **eapNoResp.** Set to TRUE by the higher layer to indicate that there will be no EAP Response for the last EAP frame delivered to EAP. This is set to FALSE by the Supplicant machine to acknowledge it has seen this variable.

b) **eapReq.** This variable is set TRUE by Supplicant Backend state machine when an EAP frame is available for processing by EAP. The higher layer will set this to FALSE when it has received the EAP frame.

c) **eapResp.** Set to TRUE by the higher layer to indicate that there is an EAP frame available for transmission to Authenticator. Set to FALSE by the Supplicant machine to indicate the EAP frame has been transmitted.

8.2.12.1.2 Constants

a) **authPeriod.** The initialization value used for the authWhile timer. Its default value is 30 s.

8.2.12.1.3 Procedures

a) **abortSupp**(). This procedure allows the Supplicant Backend state machine to release any system resources that may have been occupied during the current authentication session, prior to signaling to the Supplicant state machine that the session has been successfully aborted.

b) **getSuppRsp**(). This procedure models the processing necessary by EAP to get the information required in order to respond to the challenge specified in the most recently received EAP Request packet. This procedure does not complete until such a time as the information has been acquired; this might involve a significant delay (for example, if the challenge involves user action, such as swiping a card or typing a username and password). It should be noted that if the time taken to acquire the response information is significant, the Authenticator may time out, and repeat the request; however, the sequence number associated with the request will be the same as the original request. In the case of a repeated request, getSuppRsp returns the same information that was acquired as a result of the initial request with that sequence number.

NOTE—This can result in multiple responses being sent to the Authenticator; however, the Authentication Server will recognize, and ignore, duplicates by virtue of the sequence number that they carry being the same.

c) **txSuppRsp**(). An EAPOL frame of type EAP-Packet, containing an EAP packet from EAP, is transmitted to the Authenticator.

8.2.12.2 REQUEST

In this state, the state machine has received an EAP-Request packet from the Authenticator, and invokes EAP to perform whatever processing is needed in order to acquire the information that will form the response. Exit to the RESPONSE state occurs only after EAP has acquired this information and signaled that it has a Response packet to send. This process may involve significant processing and/or user action; there may be an unspecified delay in executing the procedures in this state. Exit to the RECEIVE state occurs if EAP has decided to ignore or discard the Authenticator's request. Exit to the FAIL state occurs if EAP has decided that the authentication has failed. Exit to the SUCCESS state occurs if EAP has decided that the authentication has succeeded.

8.2.12.3 RESPONSE

In this state, the appropriate EAP Response constructed by EAP is transmitted to the Authenticator. An unconditional exit is taken to the RECEIVE state where a timeout is started.

8.2.12.4 SUCCESS

The state machine sets the global variable suppSuccess TRUE in order to signal to the Supplicant state machine that the authentication session has terminated successfully. The global variable keyRun is also set TRUE to signal to the Supplicant Key Transmit machine that it may now run. The state machine then transitions to the IDLE state.

8.2.12.5 FAIL

The state machine sets the global variable suppFail TRUE in order to signal to the Supplicant state machine that the authentication session has terminated unsuccessfully. The state machine then transitions to the IDLE state.

8.2.12.6 TIMEOUT

The state machine sets the global variable suppTimeout TRUE in order to signal to the Supplicant state machine that the authentication session has terminated due to a timeout. The state machine then transitions to the IDLE state.

8.2.12.7 IDLE

In this state, the state machine is waiting for the Supplicant state machine to signal the start of a new authentication session. If suppStart becomes TRUE, indicating that an EAP Request packet has been received from the Authenticator, the state machine transitions to the REQUEST state in order to acquire the necessary response information from EAP. When eapSuccess becomes TRUE, the state machine transitions to the SUCCESS state. If eapFail becomes TRUE, the state machine transitions to the FAIL state.

8.2.12.8 INITIALIZE

This state is entered from any other state if a system initialization occurs, or if the Supplicant state machine sets the global variable suppAbort TRUE, signaling that the current authentication session is to be terminated. The abortSupp() procedure is used to release any system resources that may have been occupied by the session.

The state machine transitions to IDLE once the variables initialize and suppAbort are both FALSE.

NOTE—The nature of the suppAbort procedure is implementation dependent.

8.2.12.9 RECEIVE

In this state, the Supplicant is waiting for the next EAP Request from the Authenticator. A timeout is started to prevent the Supplicant from hanging if the Authenticator ceases to continue the authentication exchange. The state is entered from RESPONSE after an EAP Response is sent to the Authenticator. The state is also entered from the REQUEST state if EAP has decided to discard the previous EAP Request from the Authenticator.

9. Management of Port Access Control

This clause defines the set of managed objects, and their functionality, that allow administrative configuration and monitoring of Port Access Control.

This clause

a) Introduces the functions of management to assist in the identification of the requirements placed on Port Access Control for the support of management facilities.

b) Establishes the correspondence between the state machines used to model the operation of Port Access Control (see 8.2) and its managed objects.

c) Specifies the management operations supported by each managed object.

The management functionality specified in this clause relates to the control and monitoring of the PACP protocol and to the monitoring of session parameters; it does not include a management specification for any protocol that may be used to communicate between the Authenticator and the Authentication Server. It is assumed that the specifications associated with the chosen protocol will include specification of appropriate management functionality.

9.1 Management functions

Management functions relate to the users' needs for facilities that support the planning, organization, supervision, control, protection, and security of communications resources, and account for their use. These facilities may be categorized as supporting the functional areas of Configuration, Fault, Performance, Security, and Accounting Management. Each of these is summarized in 9.1.1 through 9.1.5, together with the facilities commonly required for the management of communication resources, and the particular facilities provided in that functional area by Port Access Control Management.

9.1.1 Configuration Management

Configuration Management provides for the identification of communications resources, initialization, reset and close-down, the supply of operational parameters, and the establishment and discovery of the relationship between resources. The facilities provided by Port Access Control Management in this functional area are as follows:

a) Configuration of the operational parameters for the Authenticator (see 9.4.1.1 and 9.4.1.2)

b) Configuration of the operational parameters for the Supplicant (see 9.5.1.1 and 9.5.1.2)

c) Configuration of the operational parameters for the System (see 9.6.1.1)

d) Initialization of the state machines for the Port (see 9.6.1.3)

9.1.2 Fault Management

Fault Management provides for fault prevention, detection, diagnosis, and correction. The facilities provided by Port Access Control Management in this functional area are as follows:

a) Retrieval of Authenticator statistical information (see 9.4.2.1)

b) Retrieval of Supplicant statistical information (see 9.5.2.1)

c) Configuration of the operational parameters for the Authenticator (see 9.4.1.1 and 9.4.1.2)

d) Configuration of the operational parameters for the Supplicant (see 9.5.1.1 and 9.5.1.2)

e) Configuration of the operational parameters for the System (see 9.6.1.1)

9.1.3 Performance Management

Performance Management provides for evaluation of the behavior of communications resources and of the effectiveness of communication activities. The facilities provided by Port Access Control Management in this functional area are as follows:

a) Retrieval of Authenticator statistical information (see 9.4.2.1)

b) Retrieval of Supplicant statistical information (see 9.5.2.1)

c) Configuration of the operational parameters for the Authenticator (see 9.4.1.1 and 9.4.1.2)

d) Configuration of the operational parameters for the Supplicant (see 9.5.1.1 and 9.5.1.2)

9.1.4 Security Management

Security Management provides for the protection of resources. The facilities provided by Port Access Control Management in this functional area are as follows:

a) Configuration of the operational parameters for the Authenticator (see 9.4.1.1 and 9.4.1.2)

b) Configuration of the operational parameters for the Supplicant (see 9.5.1.1 and 9.5.1.2)

c) Forcing reauthentication of the Supplicant (see 9.4.1.3)

9.1.5 Accounting Management

Accounting Management provides for the identification and distribution of costs and the setting of charges. The facilities provided by Port Access Control Management in this functional area is as follows:

— Retrieval of session accounting statistics (see 9.4.1.3)

9.2 Managed objects

Managed objects model the semantics of management operations. Operations upon a managed object supply information concerning, or facilitate control over, the Process or Entity associated with that managed object.

Management of Port Access Control is described in terms of the managed resources that are associated with individual Ports that support Port Access Control. The managed resources of a Port are those of the Processes and Entities established in 8.2. Specifically,

a) The state machines that support the operation of the Authenticator PAE (see 8.2.3, 8.2.4, 8.2.8, and 8.2.9). The managed objects and operations associated with these resources are defined in 8.2.

b) The state machines that support the operation of the Supplicant PAE (see 8.2.3, 8.2.11, and 8.2.12). The managed objects and operations associated with these resources are defined in 8.2.

In addition, some managed resources are not specific to the operation of an individual Authenticator PAE or Supplicant PAE, and they are therefore described as part of the overall management capability of a System and its Ports. The managed objects and operations associated with these resources are defined in 9.6.

The management of these resources is described in terms of managed objects and operations defined in 9.4, 9.5, and 9.6.

NOTE—The values specified in this clause, as inputs and outputs of management operations, are abstract information elements. Questions of formats or encoding are a matter for particular protocols that convey or otherwise represent this information.

9.3 Data types

This subclass specifies the semantics of operations independent of their encoding in management protocol. The data types of the parameters of operations are defined only as required for that specification.

The following data types are used:

a) Boolean

b) Enumerated, for a collection of named values

c) Unsigned, for all parameters specified as “the number of” some quantity

d) MAC Address

e) Time Interval, an Unsigned value representing a positive integral number of seconds, for all EAPOL protocol timeout parameters

f) Counter, for all parameters specified as a “count” of some quantity (a counter increments and wraps with a modulus of 2 to the power of 64)

9.4 Authenticator PAE managed objects

The Authenticator PAE and the state machines that support its operation are described in 8.2.3, 8.2.4, 8.2.8, and 8.2.9.

The objects that comprise this managed resource are as follows:

a) The Authenticator Configuration managed object (see 9.4.1)

b) The Authenticator Statistics managed object (see 9.4.2)

c) The Authenticator Diagnostics managed object (see 9.4.3)

d) The Authenticator Session Statistics managed object (see 9.4.4)

A Port that supports Authenticator functionality shall support the management functionality defined by the Authenticator Configuration managed object. A Port that supports Authenticator functionality may support the management functionality defined by the Authenticator Statistics, Authenticator Diagnostics, and Authenticator Session Statistics managed objects.

The means by which this management functionality is provided (e.g., the management protocol supported) shall be stated in the PICS associated with the implementation.

9.4.1 Authenticator Configuration

The Authenticator Configuration managed object models the operations that modify, or enquire about, the configuration of the Authenticator’s resources. There is a single Authenticator Configuration managed object for each Port that supports Authenticator functionality.

The management operations that can be performed on the Authenticator Configuration managed object are

a) Read Authenticator Configuration (see 9.4.1.1)

b) Set Authenticator Configuration (see 9.4.1.2)

c) Reauthenticate (see 9.4.1.3)

9.4.1.1 Read Authenticator Configuration

9.4.1.1.1 Purpose

To solicit configuration information regarding the configuration of the Authenticator associated with a Port.

9.4.1.1.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

The allocated Port Numbers are not required to be consecutive. Also, some Port Numbers may be dummy entries, with no actual LAN Port (for example, to allow for expansion of the System by addition of further MAC interfaces in the future). Such dummy Ports shall support the management operations in a manner consistent with the MAC associated with the Port being permanently disabled.

NOTE—Where Port Access Control is implemented in a MAC Bridge (see IEEE Std 802.1D), it can be convenient for the Port numbers used for Port Access Control Management to be the same as the Port numbers assigned by the Bridge. However, this is not always possible; for example, where IEEE 802.3 Link Aggregation is also implemented, Port Access Control operates on physical Ports, and the MAC Bridge makes use of the aggregated Ports.

9.4.1.1.3 Outputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **Authenticator PAE state.** The current state of the Authenticator PAE state machine (see 8.2.4). This parameter can take the following values:

1) INITIALIZE
2) DISCONNECTED
3) CONNECTING
4) AUTHENTICATING
5) AUTHENTICATED
6) ABORTING
7) HELD
8) FORCE_AUTH
9) FORCE_UNAUTH
10) RESTART

c) **Backend Authentication state.** The current state of the Backend Authentication state machine (see 8.2.9). This parameter can take the following values:

1) REQUEST
2) RESPONSE
3) SUCCESS
4) FAIL
5) TIMEOUT
6) IDLE
7) INITIALIZE
8) IGNORE

d) **AdminControlledDirections.** The current value of the AdminControlledDirections parameter associated with the Port (see 6.5). This parameter can take the following values:

1) Both
2) In

e) **OperControlledDirections.** The current value of the OperControlledDirections parameter associated with the Port (see 6.5). This parameter can take the following values:

1) Both
2) In

f) **AuthControlledPortControl.** The current value of the AuthControlledPortControl parameter associated with the Port (see 6.4). This parameter can take the following values:

1) ForceAuthorized
2) ForceUnauthorized
3) Auto

g) **AuthControlledPortStatus.** The current value of the AuthControlledPortStatus parameter associated with the Port (see 6.4). This parameter can take the following values:

1) Authorized
2) Unauthorized

h) **quietPeriod.** The value of the quietPeriod constant currently in use by the Authenticator PAE state machine (see 8.2.4.1.2).

i) **serverTimeout.** The value of the serverTimeout constant currently in use by the Backend Authentication state machine (see 8.2.9.1.2).

j) **reAuthPeriod.** The value of the reAuthPeriod constant currently in use by the Reauthentication Timer state machine (see 8.2.8.1).

k) **reAuthEnabled.** The enable/disable control used by the Reauthentication Timer state machine (see 8.2.8.1).

l) **KeyTransmissionEnabled**. TRUE if transmission of key information is enabled; FALSE if disabled (see 8.1.9).

9.4.1.2 Set Authenticator Configuration

9.4.1.2.1 Purpose

To configure the parameters that control the operation of the Authenticator associated with a Port.

9.4.1.2.2 Inputs

Any parameters marked as (optional) may be omitted from the operation to allow selective modification of a subset of the configuration parameters. However, implementations shall support the ability to include all of the following parameters:

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **AdminControlledDirections (optional).** The new value to be assigned to the AdminControlledDirections parameter associated with the Port (see 6.5). This parameter can take the following values:

1) Both
2) In

c) **AuthControlledPortControl (optional).** The new value to be assigned to the AuthControlledPortControl parameter associated with the Port (see 6.4). This parameter can take the following values:

1) ForceAuthorized

2) ForceUnauthorized

3) Auto

d) **quietPeriod (optional).** The new value to be assigned to the quietPeriod constant for the Authenticator PAE state machine (see 8.2.4.1.2).

e) **suppTimeout (optional).** The new value, in seconds, to be used to determine how long to wait for a Supplicant to respond to an EAP Request (see 8.2.9.3).

f) **serverTimeout (optional).** The new value to be assigned to the serverTimeout constant for the Backend Authentication state machine (see 8.2.9.1.2).

g) **reAuthPeriod (optional).** The new value to be assigned to the reAuthPeriod constant for the Reauthentication Timer state machine (see 8.2.8.1).

h) **reAuthEnabled (optional).** The new value to be assigned to the reAuthEnabled constant for the Reauthentication Timer state machine (see 8.2.8.1).

i) **KeyTransmissionEnabled (optional).** The new value to be assigned to the KeyTransmissionEnabled parameter (see 8.1.9).

9.4.1.2.3 Outputs

None.

9.4.1.3 Reauthenticate

9.4.1.3.1 Purpose

To cause the Authenticator PAE state machine for the Port to reauthenticate the Supplicant. If there is already an authentication in progress, this reauthentication will not occur until the current authentication completes (successfully or unsuccessfully).

9.4.1.3.2 Inputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

9.4.1.3.3 Outputs

None.

9.4.1.3.4 Effect

This operation causes the reauthenticate variable (see 8.2.2.2) for the Port's Authenticator PAE state machine to be set TRUE.

9.4.2 Authenticator Statistics

The Authenticator Statistics managed object models the operations that modify, or enquire about, the statistics associated with the operation of the Authenticator. There is a single Authenticator Statistics managed object for each Port that supports Authenticator functionality.

The management operations that can be performed on the Authenticator Statistics managed object are as follows:

— Read Authenticator Statistics (see 9.4.2.1)

9.4.2.1 Read Authenticator Statistics

9.4.2.1.1 Purpose

To solicit statistical information regarding the operation of the Authenticator associated with a Port.

9.4.2.1.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

9.4.2.1.3 Outputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **EAPOL frames received.** The number of valid EAPOL frames of any type that have been received by this Authenticator.

c) **EAPOL frames transmitted.** The number of EAPOL frames of any type that have been transmitted by this Authenticator.

d) **EAPOL Start frames received.** The number of EAPOL Start frames that have been received by this Authenticator.

e) **EAPOL Logoff frames received.** The number of EAPOL Logoff frames that have been received by this Authenticator.

f) **EAP Resp/Id frames received.** The number of valid EAP Resp/Id frames that have been received by this Authenticator.

g) **EAP Response frames received.** The number of valid EAP Response frames (other than Resp/Id frames) that have been received by this Authenticator.

h) **EAP Initial Request frames transmitted.** The number of EAP initial request frames that have been transmitted by this Authenticator.

i) **EAP Request frames transmitted.** The number of valid EAP Request frames (other than initial request frames) that have been transmitted by this Authenticator.

j) **Invalid EAPOL frames received.** The number of EAPOL frames that have been received by this Authenticator in which the frame type is not recognized.

k) **EAP length error frames received.** The number of EAPOL frames that have been received by this Authenticator in which the Packet Body Length field (see 7.5.5) is invalid.

l) **Last EAPOL frame version.** The protocol version number carried in the most recently received EAPOL frame.

m) **Last EAPOL frame source.** The source MAC address carried in the most recently received EAPOL frame.

9.4.3 Authenticator Diagnostics

The Authenticator Diagnostics managed object models the operations that modify, or enquire about, the diagnostic information associated with the operation of the Authenticator. There is a single Authenticator Diagnostics managed object for each Port that supports Authenticator functionality.

The management operations that can be performed on the Authenticator Diagnostics managed object are as follows:

— Read Authenticator Diagnostics (see 9.4.3.1)

9.4.3.1 Read Authenticator Diagnostics

9.4.3.1.1 Purpose

To solicit diagnostic information regarding the operation of the Authenticator associated with a Port.

9.4.3.1.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

9.4.3.1.3 Outputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **authEntersConnecting** (see 8.2.4.2.1 for the definition of this counter).

c) **authEapLogoffsWhileConnecting** (see 8.2.4.2.2 for the definition of this counter).

d) **authEntersAuthenticating** (see 8.2.4.2.3 for the definition of this counter).

e) **authAuthSuccessWhileAuthenticating** (see 8.2.4.2.4 for the definition of this counter).

f) **authAuthTimeoutsWhileAuthenticating** (see 8.2.4.2.5 for the definition of this counter).

g) **authAuthFailWhileAuthenticating** (see 8.2.4.2.6 for the definition of this counter).

h) **authAuthEapStartsWhileAuthenticating** (see 8.2.4.2.7 for the definition of this counter).

i) **authAuthEapLogoffWhileAuthenticating** (see 8.2.4.2.8 for the definition of this counter).

j) **authAuthReauthsWhileAuthenticated** (see 8.2.4.2.9 for the definition of this counter).

k) **authAuthEapStartsWhileAuthenticated** (see 8.2.4.2.10 for the definition of this counter).

l) **authAuthEapLogoffWhileAuthenticated** (see 8.2.4.2.11 for the definition of this counter).

m) **backendResponses** (see 8.2.9.2.1 for the definition of this counter).

n) **backendAccessChallenges** (see 8.2.9.2.2 for the definition of this counter).

o) **backendOtherRequestsToSupplicant** (see 8.2.9.2.3 for the definition of this counter).

p) **backendAuthSuccesses** (see 8.2.9.2.4 for the definition of this counter).

q) **backendAuthFails** (see 8.2.9.2.5 for the definition of this counter).

9.4.4 Authenticator Session Statistics

The Authenticator Session Statistics managed object models the operations that modify, or enquire about, the statistics associated with a Session. There is a single Authenticator Session Statistics managed object for each Port that supports Authenticator functionality.

The managed object records the statistics for the current session (if there is an active session, i.e., the portStatus variable for the Authenticator PAE state machine is set to Authorized), or the previous session (if there is no active session, i.e., the portStatus variable for the Authenticator PAE state machine is set to Unauthorized).

The management operations that can be performed on the Authenticator Session Statistics managed object are as follows:

— Read Authenticator Statistics (see 9.4.2.1)

The session statistics associated with each Port are maintained for the duration of a session, i.e., for the period of time during which the Port is authenticated. The statistics parameters are initialized, by setting their values to zero, at the point where the portStatus variable (see 8.2.2.2, 8.2.4) of the Authenticator PAE State machine transitions from Unauthorized to Authorized. While the value of portStatus remains Authorized, the session statistics are updated in accordance with their individual parameter definitions. The values of the session statistics are frozen, and not further updated, when the value of portStatus becomes Unauthorized.

NOTE—The session parameters identified here are suitable for communication to a RADIUS server at the end of a session for accounting purposes (see IETF RFC 2869); defining them in this way makes the current session parameter values available to management before the end of a session. The parameters defined may also be suitable for communication using backend authentication mechanisms supported by protocols other than RADIUS.

9.4.4.1 Read Authenticator Session Statistics

9.4.4.1.1 Purpose

To solicit statistical information regarding the current session associated with a Port.

9.4.4.1.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

9.4.4.1.3 Outputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **Session Octets Received.** The number of octets received in user data frames on this Port during the session.

c) **Session Octets Transmitted.** The number of octets transmitted in user data frames on this Port during the session.

d) **Session Frames Received.** The number of user data frames received on this Port during the session.

e) **Session Frames Transmitted.** The number of user data frames transmitted on this Port during the session.

f) **Session Identifier.** An identifier for the session, unique to this Authenticator, in the form of a printable ASCII string of at least three characters.

g) **Session Authentication Method.** The authentication method used to establish the session. This parameter can take the following values:

   1) Remote Authentication Server. The Authentication Server is external to the Authenticator's System (see 6.3).

   2) Local Authentication Server. The Authentication Server is located within the Authenticator's System.

h) **Session Time.** The duration of the session in seconds.

i) **Session Terminate Cause.** The reason for the session termination. This parameter can take the following values:

   1) Supplicant Logoff

   2) Port Failure

   3) Supplicant Restart

   4) Reauthentication Failure

   5) AuthControlledPortControl set to ForceUnauthorized

   6) Port re-initialization

   7) Port Administratively Disabled

   8) Not Terminated Yet

j) **Session User Name.** The User-Name representing the identity of the Supplicant PAE.

9.5 Supplicant PAE managed objects

The Supplicant PAE and the state machines that support its operation are described in 8.2.3, 8.2.11, and 8.2.12.

The objects that comprise this managed resource are as follows:

a) The Supplicant Configuration managed object (see 9.5.1)

b) The Supplicant Statistics managed object (see 9.5.2)

A Port that supports Supplicant functionality shall support the management functionality defined by the Supplicant Configuration managed object. A Port that supports Supplicant functionality may support the management functionality defined by the Supplicant Statistics managed object.

The means by which this management functionality is provided (e.g., the management protocol supported) shall be stated in the PICS associated with the implementation.

9.5.1 Supplicant Configuration

The Supplicant Configuration managed object models the operations that modify, or enquire about, the configuration of the Supplicant's resources. There is a single Supplicant Configuration managed object for each Port that supports Supplicant functionality.

The management operations that can be performed on the Supplicant Configuration managed object are as follows:

a) Read Supplicant Status (see 9.5.1.1)

b) Set Supplicant Configuration (see 9.5.1.2)

9.5.1.1 Read Supplicant Status

9.5.1.1.1 Purpose

To solicit configuration information regarding the configuration of the Supplicant associated with a Port.

9.5.1.1.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

The allocated Port Numbers are not required to be consecutive. Also, some Port Numbers may be dummy entries, with no actual LAN Port (for example, to allow for expansion of the System by addition of further MAC interfaces in the future). Such dummy Ports shall support the management operations in a manner consistent with the MAC associated with the Port being permanently disabled.

Where the Port is used to support the operation of a MAC Bridge Port (see IEEE Std 802.1D), the Port number used for Port Access Control Management shall be the same as the Port number assigned by the Bridge.

9.5.1.1.3 Outputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **Supplicant PAE state.** The current state of the Supplicant PAE state machine (see 8.2.11). This parameter can take the following values:

1) DISCONNECTED
2) LOGOFF
3) CONNECTING
4) AUTHENTICATING
5) AUTHENTICATED
6) Unused
7) HELD
8) RESTART
9) S_FORCE_AUTH
10) S_FORCE_UNAUTH

c) **heldPeriod.** The value of the heldPeriod constant currently in use by the Supplicant PAE state machine (see 8.2.11.1.2).

d) **authPeriod.** The value of the authPeriod constant currently in use by the Supplicant PAE state machine (see 8.2.11.1.2).

e) **startPeriod.** The value of the startPeriod constant currently in use by the Supplicant PAE state machine (see 8.2.11.1.2).

f) **maxStart.** The value of the maxStart constant currently in use by the Supplicant PAE state machine (see 8.2.11.1.2).

g) **SuppControlledPortStatus.** This directly reflects the value of the portStatus variable maintained by the Supplicant PAE state machine (see 8.2.2.2 and 8.2.11). This parameter can take the following values:
   1) Authorized
   2) Unauthorized

h) **Backend Supplicant State.** The current state of the Backend Supplicant state machine (see 8.2.12). This parameter can take the following values:
   1) INITIALIZE
   2) IDLE
   3) REQUEST
   4) RESPONSE
   5) RECEIVE
   6) FAIL
   7) SUCCESS
   8) TIMEOUT

i) **Supplicant Access Control With Authenticator.** The current state of the Supplicant Access Control With Authenticator control parameter (see 6.4, 8.2.2.2). This parameter can take the following values:
   1) Inactive
   2) Active

9.5.1.2 Set Supplicant Configuration

9.5.1.2.1 Purpose

To configure the parameters that control the operation of the Supplicant associated with a Port.

9.5.1.2.2 Inputs

Any parameters marked as (optional) may be omitted from the operation to allow selective modification of a subset of the configuration parameters. Implementations shall support the ability to include all of the parameters below.

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **heldPeriod (optional).** The new value to be assigned to the heldPeriod constant for the Supplicant PAE state machine (see 8.2.11.1.2).

c) **authPeriod (optional).** The new value to be assigned to the authPeriod constant for the Supplicant PAE state machine (see 8.2.11.1.2).

d) **startPeriod (optional).** The new value to be assigned to the startPeriod constant for the Supplicant PAE state machine (see 8.2.11.1.2).

e) **maxStart (optional).** The new value to be assigned to the maxStart constant for the Supplicant PAE state machine (see 8.2.11.1.2).

f) **Supplicant Access Control With Authenticator (optional).** The new value of the Supplicant Access Control With Authenticator control parameter (see 6.4, 8.2.2.2). This parameter can take the following values:
   1) Inactive
   2) Active

9.5.1.2.3 Outputs

None.

9.5.2 Supplicant Statistics

The Supplicant Statistics managed object models the operations that modify, or enquire about, the statistics associated with the operation of the Supplicant. There is a single Supplicant Statistics managed object for each Port that supports Supplicant functionality.

The management operations that can be performed on the Supplicant Statistics managed object are as follows:

— Read Supplicant Statistics (see 9.5.2.1)

9.5.2.1 Read Supplicant Statistics

9.5.2.1.1 Purpose

To solicit statistical information regarding the operation of the Supplicant associated with a Port.

9.5.2.1.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

9.5.2.1.3 Outputs

a) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

b) **EAPOL frames received.** The number of EAPOL frames of any type that have been received by this Supplicant.

c) **EAPOL frames transmitted.** The number of EAPOL frames of any type that have been transmitted by this Supplicant.

d) **EAPOL Start frames transmitted.** The number of EAPOL Start frames that have been transmitted by this Supplicant.

e) **EAPOL Logoff frames transmitted.** The number of EAPOL Logoff frames that have been transmitted by this Supplicant.

f) **EAP Resp/Id frames transmitted.** The number of valid EAP Resp/Id frames that have been transmitted by this Supplicant.

g) **EAP Response frames transmitted.** The number of valid EAP Response frames (other than Resp/Id frames) that have been transmitted by this Supplicant.

h) **EAP Req/Id frames received.** The number of valid EAP Req/Id frames that have been received by this Supplicant.

i) **EAP Request frames received.** The number of valid EAP Request frames (other than Req/Id frames) that have been received by this Supplicant.

j) **Invalid EAPOL frames received.** The number of EAPOL frames that have been received by this Supplicant in which the frame type is not recognized.

k) **EAP length error frames received.** The number of EAPOL frames that have been received by this Supplicant in which the Packet Body Length field (see 7.5.5) is invalid.

l) **Last EAPOL frame version.** The protocol version number carried in the most recently received EAPOL frame.

m) **Last EAPOL frame source.** The source MAC address carried in the most recently received EAPOL frame.

9.6 System managed objects

The objects that comprise this managed resource are as follows:

— The System Configuration managed object (see 9.6.1)

A Port that supports PAE functionality, in the role of a Supplicant or an Authenticator, shall support the management functionality defined by the System Configuration managed object.

The means by which this management functionality is provided (e.g., the management protocol supported) shall be stated in the PICS associated with the implementation.

9.6.1 System Configuration

The System Configuration managed object models the operations that modify, or enquire about, the configuration of the System's resources. There is a single System Configuration managed object for each System that supports Port Access Control functionality.

The management operations that can be performed on the System Configuration managed object are as follows:

a) Read System Configuration (see 9.6.1.1)

b) Set System Configuration (see 9.6.1.2)

c) Initialize Port (see 9.6.1.3)

9.6.1.1 Read System Configuration

9.6.1.1.1 Purpose

To read the configuration information associated with the System.

9.6.1.1.2 Inputs

None.

9.6.1.1.3 Outputs

a) **SystemAuthControl (Optional).** The value of the SystemAuthControl parameter (see 6.4) for the System. This parameter can take the values Enabled and Disabled. This parameter is present only in systems that support Authenticator functionality.

b) For each Port of the system:

1) **Port number.** The identification number assigned to the Port by the System in which the Port resides.

2) **Protocol version.** The protocol version number of the EAPOL implementation supported by the Port (see 7.5.3).

3) **PAE Capabilities.** The capabilities of the PAE associated with the Port. This parameter indicates whether Authenticator functionality, Supplicant functionality, both, or neither, is supported by the Port's PAE.

9.6.1.2 Set System Configuration

9.6.1.2.1 Purpose

To set the configuration information associated with the System.

9.6.1.2.2 Inputs

— **SystemAuthControl**. The desired value of the SystemAuthControl parameter (see 6.4) for the System. This parameter can take the values Enabled and Disabled.

9.6.1.2.3 Outputs

None.

9.6.1.3 Initialize Port

9.6.1.3.1 Purpose

To cause the EAPOL state machines for the Port to be initialized.

9.6.1.3.2 Inputs

— **Port number.** The identification number assigned to the Port by the System in which the Port resides.

9.6.1.3.3 Outputs

None.

9.6.1.3.4 Effect

This operation causes the initialize global variable (see 8.2.2.2) for the Port to be set TRUE for a short period of time, and then set FALSE.

NOTE—The "short period of time" for which initialize is asserted needs to be sufficiently long for all of the Port's state machines to recognize the change in state and to effect any global transitions that are required as a result. This time period is therefore implementation dependent.

10. Management protocol

10.1 Introduction

This clause defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines objects for managing the operation of Port Access Control, based on the specification contained in Clause 8 and Clause 9. This clause includes a MIB module that is SNMPv2 SMI compliant.

10.2 The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to Section 7 of IETF RFC 3410.

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in IETF STD 58, RFC 2578; IETF STD 58, RFC 2579; and IETF STD 58, RFC 2580.

10.3 Security considerations

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

SNMPv1 by itself is not a secure environment. Even if the network is secure (for example, by using IPSec), there is no control as to who on the secure network is allowed to access (read/change/create/delete) the objects in this MIB.

It is recommended that the implementors consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model, IETF RFC 3414, and the View-based Access Control Model, IETF RFC 3415, is recommended. It then becomes a user responsibility to ensure that the SNMP entity giving access to an instance of this MIB is properly configured to give access only to those principals (users) that have legitimate rights to access (change/create/delete) them, as appropriate.

10.4 Structure of the MIB

A single MIB module is defined in this clause. Objects in the MIB are arranged into groups. Each group is organized as a set of related objects. The overall structure and assignment of objects to their groups is shown in the following subclauses.

10.4.1 Relationship to the managed objects defined in Clause 9

Table 10-1 contains cross-references between the objects defined in Clause 9 and the MIB objects defined in this clause.

Table 10-1—Managed object cross-reference table

Definition in Clause 9	MIB object(s)
9.6.1 System Configuration	**dot1xPaeSystem**
Port number	dot1xPaePortNumber (table index)
SystemAuthControl	dot1xPaeSystemAuthControl
Protocol version	dot1xPaePortProtocolVersion
PAE capabilities	dot1xPaePortCapabilities
Initialize Port	dot1xPaePortInitialize
9.4.1 Authenticator Configuration	**dot1xAuthConfigTable**
Port number	dot1xPaePortNumber (table index)
Authenticator PAE State	dot1xAuthPaeState
Backend Authentication State	dot1xAuthBackendAuthState
AdminControlledDirections	dot1xAuthAdminControlledDirections
OperControlledDirections	dot1xAuthOperControlledDirections
AuthControlledPortStatus	dot1xAuthAuthControlledPortStatus
AuthControlledPortControl	dot1xAuthAuthControlledPortControl
quietPeriod	dot1xAuthQuietPeriod
suppTimeout (deprecated)	dot1xAuthSuppTimeout
serverTimeout	dot1xAuthServerTimeout
reAuthPeriod	dot1xAuthReAuthPeriod
reAuthEnabled	dot1xAuthReAuthEnabled
KeyTransmissionEnabled	dot1xAuthKeyTxEnabled
Reauthenticate	dot1xPaePortReauthenticate
9.4.2 Authenticator Statistics	**dot1xAuthStatsTable**
Port number	dot1xPaePortNumber (table index)
EAPOL frames received	dot1xAuthEapolFramesRx
EAPOL frames transmitted	dot1xAuthEapolFramesTx
EAPOL Start frames received	dot1xAuthEapolStartFramesRx
EAPOL Logoff frames received	dot1xAuthEapolLogoffFramesRx
EAPOL Resp/Id frames received	dot1xAuthEapolRespIdFramesRx
EAP Response frames received	dot1xAuthEapolRespFramesRx
EAP Initial Request frames transmitted	dot1xAuthEapolReqIdFramesTx
EAP Request frames transmitted	dot1xAuthEapolReqFramesTx

Table 10-1—Managed object cross-reference table *(continued)*

Definition in Clause 9	MIB object(s)
Invalid EAPOL frames received	dot1xAuthInvalidEapolFramesRx
EAP length error frames received	dot1xAuthEapLengthErrorFramesRx
Last EAPOL frame version	dot1xAuthLastEapolFrameVersion
Last EAPOL frame source	dot1xAuthLastEapolFrameSource
9.4.3 Authenticator Diagnostics	**dot1xAuthDiagTable**
Port number	dot1xPaePortNumber (table index)
authEntersConnecting (deprecated)	dot1xAuthEntersConnecting
authEapLogoffsWhileConnecting (deprecated)	dot1xAuthEapLogoffsWhileConnecting
authEntersAutheniticating (deprecated)	dot1xAuthEntersAuthenticating
authAuthSuccessWhileAuthenticating (deprecated)	dot1xAuthAuthSuccessWhileAuthenticating
authAuthTimeoutsWhileAuthenticating (deprecated)	dot1xAuthAuthTimeoutsWhileAuthenticating
authAuthFailWhileAuthenticating (deprecated)	dot1xAuthAuthFailWhileAuthenticating
authAuthEapStartsWhileAuthenticating (deprecated)	dot1xAuthAuthEapStartsWhileAuthenticating
authAuthLogoffWhileAuthenticating (deprecated)	dot1xAuthAuthEapLogoffWhileAuthenticating
authAuthReauthsWhileAuthenticated (deprecated)	dot1xAuthAuthReauthsWhileAuthenticated
authAuthEapStartsWhileAuthenticated (deprecated)	dot1xAuthAuthEapStartsWhileAuthenticated
authAuthLogoffWhileAuthenticated (deprecated)	dot1xAuthAuthEapLogoffWhileAuthenticated
backendResponses (deprecated)	dot1xAuthBackendResponses
backendAccessChallenges (deprecated)	dot1xAuthBackendAccessChallenges
backendOtherRequestsToSupplicant (deprecated)	dot1xAuthBackendOtherRequestsToSupplicant
backendAuthSuccesses (deprecated)	dot1xAuthBackendAuthSuccesses
backendAuthFails (deprecated)	dot1xAuthBackendAuthFails
9.4.4 Authenticator Session Statistics	**dot1xAuthSessionStatsTable**
Port number	dot1xPaePortNumber (table index)
Session Octets Received	dot1xAuthSessionOctetsRx
Session Octets Transmitted	dot1xAuthSessionOctetsTx
Session Frames Received	dot1xAuthSessionFramesRx
Session Frames Transmitted	dot1xAuthSessionFramesTx
Session Identifier	dot1xAuthSessionId
Session Authentication Method	dot1xAuthSessionAuthenticMethod
Session Time	dot1xAuthSessionTime

Table 10-1—Managed object cross-reference table *(continued)*

Definition in Clause 9	MIB object(s)
Session Terminate Cause	dot1xAuthSessionTerminateCause
Session User Name	dot1xAuthSessionUserName
9.5.1 Supplicant Configuration	**dot1xSuppConfigTable**
Port number	dot1xPaePortNumber (table index)
Supplicant PAE State	dot1xSuppPaeState
heldPeriod	dot1xSuppHeldPeriod
authPeriod	dot1xSuppAuthPeriod
startPeriod	dot1xSuppStartPeriod
maxStart	dot1xSuppMaxStart
SuppControlledPortStatus	dot1xSuppSuppControlledPortStatus
Supplicant Backend PAE State	dot1xSuppBackendPaeState
Supplicant Access Control With Authenticator	dot1xSuppAccessCtrlWithAuth
9.5.2 Supplicant Statistics	**dot1xSuppStatsTable**
Port number	dot1xPaePortNumber (table index)
EAPOL frames received	dot1xSuppEapolFramesRx
EAPOL frames transmitted	dot1xSuppEapolFramesTx
EAPOL Start frames transmitted	dot1xSuppEapolStartFramesTx
EAPOL Logoff frames transmitted	dot1xSuppEapolLogoffFramesTx
EAP Resp/ID frames transmitted (deprecated)	dot1xSuppEapolRespIdFramesTx
EAP Response frames transmitted (deprecated)	dot1xSuppEapolRespFramesTx
EAP Initial Request frames received (deprecated)	dot1xSuppEapolReqIdFramesRx
EAP Request frames received (deprecated)	dot1xSuppEapolReqFramesRx
Invalid EAPOL frames received	dot1xSuppInvalidEapolFramesRx
EAPOL length error frames received	dot1xSuppEapLengthErrorFramesRx
Last EAPOL frame version	dot1xSuppLastEapolFrameVersion
Last EAPOL frame source	dot1xSuppLastEapolFrameSource

10.4.2 The PAE System Group

This group of objects provides management functionality that is not specific to the operation of either of the two PAE roles (Supplicant and Authenticator). A means of enabling and disabling the operation of Port Access Control for the entire system is provided, plus a per-Port indication of the protocol version supported and the PAE roles supported by the port. As it is not mandatory for all Ports of a System to support PAE

functionality, there may be Port entries that indicate Ports that support neither Supplicant nor Authenticator functionality.

10.4.3 The PAE Authenticator Group

This group of objects provides, for each Port of a System, the functionality necessary to allow configuration of the operation of the Authenticator PAE, recording and retrieving statistical information relating to the operation of the Authenticator PAE, and recording and retrieving information relating to a session (i.e., the period of time between consecutive authentications on the Port).

10.4.4 The PAE Supplicant Group

This group of objects provides, for each Port of a System, the functionality necessary to allow configuration of the operation of the Supplicant PAE, and recording and retrieving statistical information relating to the operation of the Authenticator PAE.

10.5 Relationship to other MIBs

It is assumed that a system implementing this MIB will also implement (at least) the "system" group defined in MIB-II defined in IETF RFC 3418 and the "interfaces" group defined in IETF RFC 2863.

10.5.1 Relationship to the Interfaces MIB

IETF RFC 2863, the Interface MIB Evolution, requires that any MIB that is an adjunct of the Interface MIB clarify specific areas within the Interface MIB. These areas were intentionally left vague in IETF RFC 2863 to avoid over constraining the MIB, thereby precluding management of certain media types.

Section 3.3 of IETF RFC 2863 enumerates several areas that a media-specific MIB must clarify. Each of these areas is addressed in a following subsection. The implementor is referred to IETF RFC 2863 in order to understand the general intent of these areas.

In IETF RFC 2863, the "interfaces" group is defined as being mandatory for all systems and contains information on an entity's interfaces, where each interface is thought of as being attached to a subnetwork. (Note that this term is not to be confused with subnet, which refers to an addressing partitioning scheme used in the Internet suite of protocols.) The term *segment* is sometimes used to refer to such a subnetwork.

Where Port numbers are used in this standard to identify Ports of a System, these numbers are equal to the ifIndex value for the interface for the corresponding Port.

10.6 Definitions for Port Access Control MIB[13]

In the MIB definition below, should any discrepancy between the DESCRIPTION text and the corresponding definition in Clause 9 occur, the definition in Clause 9 shall take precedence.

```
IEEE8021-PAE-MIB DEFINITIONS ::= BEGIN

-- ------------------------------------------------------------ --
-- IEEE 802.1X MIB
-- ------------------------------------------------------------ --

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Counter64,
    Unsigned32, TimeTicks
        FROM SNMPv2-SMI
    MacAddress, TEXTUAL-CONVENTION, TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    InterfaceIndex
        FROM IF-MIB
    ;

ieee8021paeMIB MODULE-IDENTITY
    LAST-UPDATED "200406220000Z"
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "http://grouper.ieee.org/groups/802/1/index.html"

    DESCRIPTION
        "The Port Access Entity module for managing IEEE
        802.1X."

    REVISION     "200406220000Z"  -- June 22nd, 2004
    DESCRIPTION
        "IEEE Std. 802.1X-2004 revision:
          - In the MODULE-IDENTITY value assignment changed
            'iso(1)' to 'iso';
          - Clarified original references to 802.1X-2001;
          - Added references to 802.1X-2004;
          - Added restart(10) to dot1xAuthPaeState;
          - Added ignore(8) to dot1xAuthBackendAuthState;
          - Deprecated dot1xAuthTxPeriod, dot1xSuppTimeout,
            dotxAuthMaxReq, all of dot1xAuthDiagTable,
            dot1xSuppEapolRespIdFramesTx,
            dot1xSuppEapolRespFramesTx,
            dot1xSuppEapolReqIdFramesRx,
            dot1xSuppEapolReqFramesRx;
          - Added restart(8), sForceAuth(9) and
```

[13]*Copyright release for MIBs:* Users of this standard may freely reproduce the MIB definition contained in this clause so that it can be used for its intended purpose.


```
            sForceUnauth(10) to dot1xSuppPaeState;
          - Added dot1xSuppControlledPortStatus;
          - Added dot1xSuppAccessCtrlWithAuth;
          - Added dot1xSuppBackendState;
          - Bug fix to add dot1xPaePortReauthenticate and
            dot1xAuthSessionUserName to the appropriate
            conformance groups;
          - Updated conformance groups for new and deprecated
            objects;
          - Deprecated dot1xPaeCompliance;
          - Added dot1xPaeCompliance2."

    REVISION     "200101160000Z"  -- Jan 16th, 2001
    DESCRIPTION
        "IEEE Std. 802.1X-2001 initial version."

    ::= { iso std(0) iso8802(8802) ieee802dot1(1)
          ieee802dot1mibs(1) 1 }

paeMIBObjects OBJECT IDENTIFIER ::= { ieee8021paeMIB 1 }

-- ------------------------------------------------------------ --
-- Textual Conventions
-- ------------------------------------------------------------ --

PaeControlledDirections ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The control mode values for the Authenticator PAE."
    SYNTAX      INTEGER {
                    both(0),
                    in(1)
                }

PaeControlledPortStatus ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The status values of the Authenticator PAE controlled
        Port."
    SYNTAX      INTEGER {
                    authorized(1),
                    unauthorized(2)
                }

PaeControlledPortControl ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The control values of the Authenticator PAE controlled
        Port."
    SYNTAX      INTEGER {
                    forceUnauthorized(1),
                    auto(2),
                    forceAuthorized(3)
                }
```

```
-- ------------------------------------------------------- --

-- ------------------------------------------------------- --
-- groups in the PAE MIB
-- ------------------------------------------------------- --

dot1xPaeSystem        OBJECT IDENTIFIER ::= { paeMIBObjects 1 }
dot1xPaeAuthenticator OBJECT IDENTIFIER ::= { paeMIBObjects 2 }
dot1xPaeSupplicant    OBJECT IDENTIFIER ::= { paeMIBObjects 3 }

-- ------------------------------------------------------- --

-- ------------------------------------------------------- --
-- The PAE System Group
-- ------------------------------------------------------- --

dot1xPaeSystemAuthControl OBJECT-TYPE
    SYNTAX      INTEGER { enabled(1), disabled(2) }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative enable/disable state for
         Port Access Control in a System."
    REFERENCE
        "802.1X-2001 9.6.1, SystemAuthControl,
         802.1X-2004 9.6.1, SystemAuthControl"
    ::= { dot1xPaeSystem 1 }

-- ------------------------------------------------------- --
-- The PAE Port Table
-- ------------------------------------------------------- --

dot1xPaePortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1xPaePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of system level information for each port
        supported by the Port Access Entity.  An entry appears
        in this table for each port of this system."
    REFERENCE
        "802.1X-2001 9.6.1,
         802.1X-2004 9.6.1"
    ::= { dot1xPaeSystem 2 }

dot1xPaePortEntry OBJECT-TYPE
    SYNTAX      Dot1xPaePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Port number, protocol version, and
        initialization control for a Port."
    INDEX { dot1xPaePortNumber }
```

```
    ::= { dot1xPaePortTable 1 }

Dot1xPaePortEntry ::=
    SEQUENCE {
        dot1xPaePortNumber
            InterfaceIndex,
        dot1xPaePortProtocolVersion
            Unsigned32,
        dot1xPaePortCapabilities
            BITS,
        dot1xPaePortInitialize
            TruthValue,
        dot1xPaePortReauthenticate
            TruthValue
        }

dot1xPaePortNumber OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Port number associated with this Port."
    REFERENCE
        "802.1X-2001 9.6.1, Port number,
         802.1X-2004 9.6.1, Port number"
    ::= { dot1xPaePortEntry 1 }

dot1xPaePortProtocolVersion OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The protocol version associated with this Port."
    REFERENCE
        "802.1X-2001 9.6.1, Protocol version,
         802.1X-2004 9.6.1, Protocol version"
    ::= { dot1xPaePortEntry 2 }

dot1xPaePortCapabilities OBJECT-TYPE
    SYNTAX      BITS {
                    dot1xPaePortAuthCapable(0),
                        -- Authenticator functions are supported
                    dot1xPaePortSuppCapable(1)
                        -- Supplicant functions are supported
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the PAE functionality that this Port
        supports and that may be managed through this MIB."
    REFERENCE
        "802.1X-2001 9.6.1, PAE Capabilities,
         802.1X-2004 9.6.1, PAE Capabilities"
    ::= { dot1xPaePortEntry 3 }
```

```
dot1xPaePortInitialize OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The initialization control for this Port. Setting this
        attribute TRUE causes the Port to be initialized.
        The attribute value reverts to FALSE once initialization
        has completed."
    REFERENCE
        "802.1X-2001 9.6.1.3, Initialize Port,
         802.1X-2004 9.6.1.3, Initialize Port"
    ::= { dot1xPaePortEntry 4 }

dot1xPaePortReauthenticate OBJECT-TYPE
    SYNTAX TruthValue
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "The reauthentication control for this port. Setting
        this attribute TRUE causes the Authenticator PAE state
        machine for the Port to reauthenticate the Supplicant.
        Setting this attribute FALSE has no effect.
        This attribute always returns FALSE when it is read."
    REFERENCE
        "802.1X-2001 9.4.1.3 Reauthenticate,
         802.1X-2004 9.4.1.3 Reauthenticate"
    ::= { dot1xPaePortEntry 5 }

-- ---------------------------------------------------------- --
-- The PAE Authenticator Group
-- ---------------------------------------------------------- --

-- ---------------------------------------------------------- --
-- The Authenticator Configuration Table
-- ---------------------------------------------------------- --

dot1xAuthConfigTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1xAuthConfigEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains the configuration objects for the
        Authenticator PAE associated with each port.
        An entry appears in this table for each port that may
        authenticate access to itself."
    REFERENCE
        "802.1X-2001 9.4.1 Authenticator Configuration,
         802.1X-2004 9.4.1 Authenticator Configuration"
    ::= { dot1xPaeAuthenticator 1 }

dot1xAuthConfigEntry OBJECT-TYPE
    SYNTAX      Dot1xAuthConfigEntry
```



```
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The configuration information for an Authenticator
            PAE."
        INDEX { dot1xPaePortNumber }
        ::= { dot1xAuthConfigTable 1 }

    Dot1xAuthConfigEntry ::=
        SEQUENCE {
            dot1xAuthPaeState
                INTEGER,
            dot1xAuthBackendAuthState
                INTEGER,
            dot1xAuthAdminControlledDirections
                PaeControlledDirections,
            dot1xAuthOperControlledDirections
                PaeControlledDirections,
            dot1xAuthAuthControlledPortStatus
                PaeControlledPortStatus,
            dot1xAuthAuthControlledPortControl
                PaeControlledPortControl,
            dot1xAuthQuietPeriod
                Unsigned32,
            dot1xAuthTxPeriod
                Unsigned32,
            dot1xAuthSuppTimeout
                Unsigned32,
            dot1xAuthServerTimeout
                Unsigned32,
            dot1xAuthMaxReq
                Unsigned32,
            dot1xAuthReAuthPeriod
                Unsigned32,
            dot1xAuthReAuthEnabled
                TruthValue,
            dot1xAuthKeyTxEnabled
                TruthValue
            }

    dot1xAuthPaeState OBJECT-TYPE
        SYNTAX      INTEGER {
                        initialize(1),
                        disconnected(2),
                        connecting(3),
                        authenticating(4),
                        authenticated(5),
                        aborting(6),
                        held(7),
                        forceAuth(8),
                        forceUnauth(9),
                        restart(10)
                    }
        MAX-ACCESS  read-only
```

```
        STATUS      current
        DESCRIPTION
            "The current value of the Authenticator PAE state
            machine."
        REFERENCE
            "802.1X-2001 9.4.1, Authenticator PAE state,
             802.1X-2004 9.4.1, Authenticator PAE state"
        ::= { dot1xAuthConfigEntry 1 }

    dot1xAuthBackendAuthState OBJECT-TYPE
        SYNTAX      INTEGER {
                        request(1),
                        response(2),
                        success(3),
                        fail(4),
                        timeout(5),
                        idle(6),
                        initialize(7),
                        ignore(8)
                    }
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The current state of the Backend Authentication
            state machine."
        REFERENCE
            "802.1X-2001 9.4.1, Backend Authentication state,
             802.1X-2004 9.4.1, Backend Authentication state"
        ::= { dot1xAuthConfigEntry 2 }

    dot1xAuthAdminControlledDirections OBJECT-TYPE
        SYNTAX      PaeControlledDirections
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "The current value of the administrative controlled
            directions parameter for the Port."
        REFERENCE
            "802.1X-2001 9.4.1, Admin Control Mode,
             802.1X-2004 9.4.1, Admin Control Mode"
        ::= { dot1xAuthConfigEntry 3 }

    dot1xAuthOperControlledDirections OBJECT-TYPE
        SYNTAX      PaeControlledDirections
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The current value of the operational controlled
            directions parameter for the Port."
        REFERENCE
            "802.1X-2001 9.4.1, Oper Control Mode,
             802.1X-2004 9.4.1, Oper Control Mode"
        ::= { dot1xAuthConfigEntry 4 }
```

```
dot1xAuthAuthControlledPortStatus OBJECT-TYPE
    SYNTAX      PaeControlledPortStatus
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current value of the controlled Port
        status parameter for the Port."
    REFERENCE
        "802.1X-2001 9.4.1, AuthControlledPortStatus,
         802.1X-2004 9.4.1, AuthControlledPortStatus"
    ::= { dot1xAuthConfigEntry 5 }

dot1xAuthAuthControlledPortControl OBJECT-TYPE
    SYNTAX      PaeControlledPortControl
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The current value of the controlled Port
        control parameter for the Port."
    REFERENCE
        "802.1X-2001 9.4.1, AuthControlledPortControl,
         802.1X-2004 9.4.1, AuthControlledPortControl"
    ::= { dot1xAuthConfigEntry 6 }

dot1xAuthQuietPeriod OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value, in seconds, of the quietPeriod constant
        currently in use by the Authenticator PAE state
        machine."
    REFERENCE
        "802.1X-2001 9.4.1, quietPeriod,
         802.1X-2004 9.4.1, quietPeriod"
    DEFVAL { 60 }
    ::= { dot1xAuthConfigEntry 7 }

dot1xAuthTxPeriod OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      deprecated
    DESCRIPTION
        "The value, in seconds, of the txPeriod constant
        currently in use by the Authenticator PAE state
        machine."
    REFERENCE
        "802.1X-2001 9.4.1, txPeriod"
    DEFVAL { 30 }
    ::= { dot1xAuthConfigEntry 8 }

dot1xAuthSuppTimeout OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
```

```
        STATUS      deprecated
        DESCRIPTION
            "The value, in seconds, of the suppTimeout constant
            currently in use by the Backend Authentication state
            machine."
        REFERENCE
            "802.1X-2001 9.4.1, suppTimeout,
             802.1X-2004 9.4.1, suppTimeout"
        DEFVAL { 30 }
        ::= { dot1xAuthConfigEntry 9 }

    dot1xAuthServerTimeout OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "The value, in seconds, of the serverTimeout constant
            currently in use by the Backend Authentication state
            machine."
        REFERENCE
            "802.1X-2001 9.4.1, serverTimeout,
             802.1X-2004 9.4.1, serverTimeout"
        DEFVAL { 30 }
        ::= { dot1xAuthConfigEntry 10 }

    dot1xAuthMaxReq OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-write
        STATUS      deprecated
        DESCRIPTION
            "The value of the maxReq constant currently in use by
            the Backend Authentication state machine."
        REFERENCE
            "802.1X-2001 9.4.1, maxReq"
        DEFVAL { 2 }
        ::= { dot1xAuthConfigEntry 11 }

    dot1xAuthReAuthPeriod OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "The value, in seconds, of the reAuthPeriod constant
            currently in use by the Reauthentication Timer state
            machine."
        REFERENCE
            "802.1X-2001 9.4.1, reAuthPeriod,
             802.1X-2004 9.4.1, reAuthPeriod"
        DEFVAL { 3600 }
        ::= { dot1xAuthConfigEntry 12 }

    dot1xAuthReAuthEnabled OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-write
```



```
        STATUS      current
        DESCRIPTION
            "The enable/disable control used by the Reauthentication
            Timer state machine (8.5.5.1)."
        REFERENCE
            "802.1X-2001 9.4.1, reAuthEnabled,
             802.1X-2004 9.4.1, reAuthEnabled"
        DEFVAL { false }
        ::= { dot1xAuthConfigEntry 13 }

    dot1xAuthKeyTxEnabled OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "The value of the keyTransmissionEnabled constant
            currently in use by the Authenticator PAE state
            machine."
        REFERENCE
            "802.1X-2001 9.4.1, keyTransmissionEnabled,
             802.1X-2004 9.4.1, keyTransmissionEnabled"
        ::= { dot1xAuthConfigEntry 14 }

    -- ---------------------------------------------------------- --
    -- The Authenticator Statistics Table
    -- ---------------------------------------------------------- --

    dot1xAuthStatsTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Dot1xAuthStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table that contains the statistics objects for the
             Authenticator PAE associated with each Port.
             An entry appears in this table for each port that may
             authenticate access to itself."
        REFERENCE
            "802.1X-2001 9.4.2 Authenticator Statistics,
             802.1X-2004 9.4.2 Authenticator Statistics"
        ::= { dot1xPaeAuthenticator 2 }

    dot1xAuthStatsEntry OBJECT-TYPE
        SYNTAX      Dot1xAuthStatsEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The statistics information for an Authenticator PAE."
        INDEX { dot1xPaePortNumber }
        ::= { dot1xAuthStatsTable 1 }

    Dot1xAuthStatsEntry ::=
        SEQUENCE {
            dot1xAuthEapolFramesRx
                Counter32,
```

```
        dot1xAuthEapolFramesTx
            Counter32,
        dot1xAuthEapolStartFramesRx
            Counter32,
        dot1xAuthEapolLogoffFramesRx
            Counter32,
        dot1xAuthEapolRespIdFramesRx
            Counter32,
        dot1xAuthEapolRespFramesRx
            Counter32,
        dot1xAuthEapolReqIdFramesTx
            Counter32,
        dot1xAuthEapolReqFramesTx
            Counter32,
        dot1xAuthInvalidEapolFramesRx
            Counter32,
        dot1xAuthEapLengthErrorFramesRx
            Counter32,
        dot1xAuthLastEapolFrameVersion
            Unsigned32,
        dot1xAuthLastEapolFrameSource
            MacAddress
        }

dot1xAuthEapolFramesRx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of valid EAPOL frames of any type
        that have been received by this Authenticator."
    REFERENCE
        "802.1X-2001 9.4.2, EAPOL frames received,
         802.1X-2004 9.4.2, EAPOL frames received"
    ::= { dot1xAuthStatsEntry 1 }

dot1xAuthEapolFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL frames of any type
        that have been transmitted by this Authenticator."
    REFERENCE
        "802.1X-2001 9.4.2, EAPOL frames transmitted,
         802.1X-2004 9.4.2, EAPOL frames transmitted"
    ::= { dot1xAuthStatsEntry 2 }

dot1xAuthEapolStartFramesRx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL Start frames that have
```



```
            been received by this Authenticator."
        REFERENCE
            "802.1X-2001 9.4.2, EAPOL Start frames received,
             802.1X-2004 9.4.2, EAPOL Start frames received"
        ::= { dot1xAuthStatsEntry 3 }

    dot1xAuthEapolLogoffFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of EAPOL Logoff frames that have
            been received by this Authenticator."
        REFERENCE
            "802.1X-2001 9.4.2, EAPOL Logoff frames received,
             802.1X-2004 9.4.2, EAPOL Logoff frames received"
        ::= { dot1xAuthStatsEntry 4 }

    dot1xAuthEapolRespIdFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of EAP Resp/Id frames that have
            been received by this Authenticator."
        REFERENCE
            "802.1X-2001 9.4.2, EAPOL Resp/Id frames received,
             802.1X-2004 9.4.2, EAPOL Resp/Id frames received"
        ::= { dot1xAuthStatsEntry 5 }

    dot1xAuthEapolRespFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of valid EAP Response frames
            (other than Resp/Id frames) that have been
            received by this Authenticator."
        REFERENCE
            "802.1X-2001 9.4.2, EAPOL Response frames received,
             802.1X-2004 9.4.2, EAPOL Response frames received"
        ::= { dot1xAuthStatsEntry 6 }

    dot1xAuthEapolReqIdFramesTx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of EAP Req/Id frames that have been
            transmitted by this Authenticator."
        REFERENCE
            "802.1X-2001 9.4.2, EAPOL Req/Id frames transmitted,
             802.1X-2004 9.4.2, EAPOL Req/Id frames transmitted"
        ::= { dot1xAuthStatsEntry 7 }
```

```
dot1xAuthEapolReqFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAP Request frames
        (other than Rq/Id frames) that have been
        transmitted by this Authenticator."
    REFERENCE
        "802.1X-2001 9.4.2, EAPOL Request frames transmitted,
         802.1X-2004 9.4.2, EAPOL Request frames transmitted"
    ::= { dot1xAuthStatsEntry 8 }

dot1xAuthInvalidEapolFramesRx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL frames that have been
        received by this Authenticator in which the
        frame type is not recognized."
    REFERENCE
        "802.1X-2001 9.4.2, Invalid EAPOL frames received,
         802.1X-2004 9.4.2, Invalid EAPOL frames received"
    ::= { dot1xAuthStatsEntry 9 }

dot1xAuthEapLengthErrorFramesRx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL frames that have been received
        by this Authenticator in which the Packet Body
        Length field is invalid."
    REFERENCE
        "802.1X-2001 9.4.2, EAP length error frames received,
         802.1X-2004 9.4.2, EAP length error frames received"
    ::= { dot1xAuthStatsEntry 10 }

dot1xAuthLastEapolFrameVersion OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The protocol version number carried in the
        most recently received EAPOL frame."
    REFERENCE
        "802.1X-2001 9.4.2, Last EAPOL frame version,
         802.1X-2004 9.4.2, Last EAPOL frame version"
    ::= { dot1xAuthStatsEntry 11 }

dot1xAuthLastEapolFrameSource OBJECT-TYPE
    SYNTAX      MacAddress
```



```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The source MAC address carried in the
            most recently received EAPOL frame."
        REFERENCE
            "802.1X-2001 9.4.2, Last EAPOL frame source,
             802.1X-2004 9.4.2, Last EAPOL frame source"
        ::= { dot1xAuthStatsEntry 12 }

    -- ---------------------------------------------------------- --
    -- The Authenticator Diagnostics Table
    -- ---------------------------------------------------------- --

    dot1xAuthDiagTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Dot1xAuthDiagEntry
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
            "A table that contains the diagnostics objects for the
             Authenticator PAE associated with each Port.
             An entry appears in this table for each port that may
             authenticate access to itself."
        REFERENCE
            "802.1X-2001 9.4.3 Authenticator Diagnostics"
        ::= { dot1xPaeAuthenticator 3 }

    dot1xAuthDiagEntry OBJECT-TYPE
        SYNTAX      Dot1xAuthDiagEntry
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
            "The diagnostics information for an Authenticator PAE."
        INDEX { dot1xPaePortNumber }
        ::= { dot1xAuthDiagTable 1 }

    Dot1xAuthDiagEntry ::=
        SEQUENCE {
            dot1xAuthEntersConnecting
                Counter32,
            dot1xAuthEapLogoffsWhileConnecting
                Counter32,
            dot1xAuthEntersAuthenticating
                Counter32,
            dot1xAuthAuthSuccessWhileAuthenticating
                Counter32,
            dot1xAuthAuthTimeoutsWhileAuthenticating
                Counter32,
            dot1xAuthAuthFailWhileAuthenticating
                Counter32,
            dot1xAuthAuthReauthsWhileAuthenticating
                Counter32,
            dot1xAuthAuthEapStartsWhileAuthenticating
                Counter32,
```

```
        dot1xAuthAuthEapLogoffWhileAuthenticating
            Counter32,
        dot1xAuthAuthReauthsWhileAuthenticated
            Counter32,
        dot1xAuthAuthEapStartsWhileAuthenticated
            Counter32,
        dot1xAuthAuthEapLogoffWhileAuthenticated
            Counter32,
        dot1xAuthBackendResponses
            Counter32,
        dot1xAuthBackendAccessChallenges
            Counter32,
        dot1xAuthBackendOtherRequestsToSupplicant
            Counter32,
        dot1xAuthBackendNonNakResponsesFromSupplicant
            Counter32,
        dot1xAuthBackendAuthSuccesses
            Counter32,
        dot1xAuthBackendAuthFails
            Counter32
        }

dot1xAuthEntersConnecting OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions to the CONNECTING state from any other
        state."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.1"
    ::= { dot1xAuthDiagEntry 1 }

dot1xAuthEapLogoffsWhileConnecting OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from CONNECTING to DISCONNECTED as a result
        of receiving an EAPOL-Logoff message."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.2"
    ::= { dot1xAuthDiagEntry 2 }

dot1xAuthEntersAuthenticating OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from CONNECTING to AUTHENTICATING, as a
        result of an EAP-Response/Identity message being
```



```
            received from the Supplicant."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.3"
        ::= { dot1xAuthDiagEntry 3 }

    dot1xAuthAuthSuccessWhileAuthenticating OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            transitions from AUTHENTICATING to AUTHENTICATED, as a
            result of the Backend Authentication state machine
            indicating successful authentication of the Supplicant
            (authSuccess = TRUE)."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.4"
        ::= { dot1xAuthDiagEntry 4 }

    dot1xAuthAuthTimeoutsWhileAuthenticating OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            transitions from AUTHENTICATING to ABORTING, as a result
            of the Backend Authentication state machine indicating
            authentication timeout (authTimeout = TRUE)."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.5"
        ::= { dot1xAuthDiagEntry 5 }

    dot1xAuthAuthFailWhileAuthenticating OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            transitions from AUTHENTICATING to HELD, as a result
            of the Backend Authentication state machine indicating
            authentication failure (authFail = TRUE)."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.6"
        ::= { dot1xAuthDiagEntry 6 }

    dot1xAuthAuthReauthsWhileAuthenticating OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            transitions from AUTHENTICATING to ABORTING, as a result
            of a reauthentication request (reAuthenticate = TRUE)."
        REFERENCE
```

```
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.7"
    ::= { dot1xAuthDiagEntry 7 }

dot1xAuthAuthEapStartsWhileAuthenticating OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from AUTHENTICATING to ABORTING, as a result
        of an EAPOL-Start message being received
        from the Supplicant."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.8"
    ::= { dot1xAuthDiagEntry 8 }

dot1xAuthAuthEapLogoffWhileAuthenticating OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from AUTHENTICATING to ABORTING, as a result
        of an EAPOL-Logoff message being received
        from the Supplicant."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.9"
    ::= { dot1xAuthDiagEntry 9 }

dot1xAuthAuthReauthsWhileAuthenticated OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from AUTHENTICATED to CONNECTING, as a
        result of a reauthentication request
        (reAuthenticate = TRUE)."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.10"
    ::= { dot1xAuthDiagEntry 10 }

dot1xAuthAuthEapStartsWhileAuthenticated OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from AUTHENTICATED to CONNECTING, as a
        result of an EAPOL-Start message being received from the
        Supplicant."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.11"
    ::= { dot1xAuthDiagEntry 11 }
```



```
dot1xAuthAuthEapLogoffWhileAuthenticated OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        transitions from AUTHENTICATED to DISCONNECTED, as a
        result of an EAPOL-Logoff message being received from
        the Supplicant."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.4.2.12"
    ::= { dot1xAuthDiagEntry 12 }

dot1xAuthBackendResponses OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine sends
        an initial Access-Request packet to the Authentication
        server (i.e., executes sendRespToServer on entry to the
        RESPONSE state). Indicates that the Authenticator
        attempted communication with the Authentication Server."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.6.2.1"
    ::= { dot1xAuthDiagEntry 13 }

dot1xAuthBackendAccessChallenges OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        receives an initial Access-Challenge packet from the
        Authentication server (i.e., aReq becomes TRUE,
        causing exit from the RESPONSE state). Indicates that
        the Authentication Server has communication with
        the Authenticator."
    REFERENCE
        "802.1X-2001 9.4.2, 802.1X-2001 8.5.6.2.2"
    ::= { dot1xAuthDiagEntry 14 }

dot1xAuthBackendOtherRequestsToSupplicant OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "Counts the number of times that the state machine
        sends an EAP-Request packet (other than an Identity,
        Notification, Failure or Success message) to the
        Supplicant (i.e., executes txReq on entry to the
        REQUEST state). Indicates that the Authenticator chose
        an EAP-method."
```

```
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.6.2.3"
        ::= { dot1xAuthDiagEntry 15 }

    dot1xAuthBackendNonNakResponsesFromSupplicant OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            receives a response from the Supplicant to an initial
            EAP-Request, and the response is something other than
            EAP-NAK (i.e., rxResp becomes TRUE, causing the state
            machine to transition from REQUEST to RESPONSE,
            and the response is not an EAP-NAK). Indicates that
            the Supplicant can respond to the Authenticator's
            chosen EAP-method."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.6.2.4"
        ::= { dot1xAuthDiagEntry 16 }

    dot1xAuthBackendAuthSuccesses OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            receives an EAP-Success message from the Authentication
            Server (i.e., aSuccess becomes TRUE, causing a
            transition from RESPONSE to SUCCESS). Indicates that
            the Supplicant has successfully authenticated to
            the Authentication Server."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.6.2.5"
        ::= { dot1xAuthDiagEntry 17 }

    dot1xAuthBackendAuthFails OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "Counts the number of times that the state machine
            receives an EAP-Failure message from the Authentication
            Server (i.e., aFail becomes TRUE, causing a transition
            from RESPONSE to FAIL). Indicates that the Supplicant
            has not authenticated to the Authentication Server."
        REFERENCE
            "802.1X-2001 9.4.2, 802.1X-2001 8.5.6.2.6"
        ::= { dot1xAuthDiagEntry 18 }

    -- ------------------------------------------------------- --
    -- The Authenticator Session Statistics Table
    -- ------------------------------------------------------- --
```

```
dot1xAuthSessionStatsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1xAuthSessionStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains the session statistics objects
        for the Authenticator PAE associated with each Port.
        An entry appears in this table for each port that may
        authenticate access to itself."
    REFERENCE
        "802.1X-2001 9.4.4,
         802.1X-2004 9.4.4"
    ::= { dot1xPaeAuthenticator 4 }

dot1xAuthSessionStatsEntry OBJECT-TYPE
    SYNTAX      Dot1xAuthSessionStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The session statistics information for an Authenticator
        PAE.  This shows the current values being collected for
        each session that is still in progress, or the final
        values for the last valid session on each port where
        there is no session currently active."
    INDEX { dot1xPaePortNumber }
    ::= { dot1xAuthSessionStatsTable 1 }

Dot1xAuthSessionStatsEntry ::=
    SEQUENCE {
        dot1xAuthSessionOctetsRx
            Counter64,
        dot1xAuthSessionOctetsTx
            Counter64,
        dot1xAuthSessionFramesRx
            Counter32,
        dot1xAuthSessionFramesTx
            Counter32,
        dot1xAuthSessionId
            SnmpAdminString,
        dot1xAuthSessionAuthenticMethod
            INTEGER,
        dot1xAuthSessionTime
            TimeTicks,
        dot1xAuthSessionTerminateCause
            INTEGER,
        dot1xAuthSessionUserName
            SnmpAdminString
    }

dot1xAuthSessionOctetsRx OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

```
            "The number of octets received in user data
            frames on this Port during the session."
        REFERENCE
            "802.1X-2001 9.4.4, Session Octets Received,
             802.1X-2004 9.4.4, Session Octets Received"
        ::= { dot1xAuthSessionStatsEntry 1 }

    dot1xAuthSessionOctetsTx OBJECT-TYPE
        SYNTAX      Counter64
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of octets transmitted in user data
            frames on this Port during the session."
        REFERENCE
            "802.1X-2001 9.4.4, Session Octets Transmitted,
             802.1X-2004 9.4.4, Session Octets Transmitted"
        ::= { dot1xAuthSessionStatsEntry 2 }

    dot1xAuthSessionFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of user data frames received
            on this Port during the session."
        REFERENCE
            "802.1X-2001 9.4.4, Session Frames Received,
             802.1X-2004 9.4.4, Session Frames Received"
        ::= { dot1xAuthSessionStatsEntry 3 }

    dot1xAuthSessionFramesTx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of user data frames transmitted
            on this Port during the session."
        REFERENCE
            "802.1X-2001 9.4.4, Session Frames Transmitted,
             802.1X-2004 9.4.4, Session Frames Transmitted"
        ::= { dot1xAuthSessionStatsEntry 4 }

    dot1xAuthSessionId OBJECT-TYPE
        SYNTAX      SnmpAdminString
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "A unique identifier for the session, in the
            form of a printable ASCII string of at least
            three characters."
        REFERENCE
            "802.1X-2001 9.4.4, Session Identifier,
             802.1X-2004 9.4.4, Session Identifier"
```



```
    ::= { dot1xAuthSessionStatsEntry 5 }

dot1xAuthSessionAuthenticMethod OBJECT-TYPE
    SYNTAX      INTEGER {
                    remoteAuthServer(1),
                    localAuthServer(2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The authentication method used to establish the
        session."
    REFERENCE
        "802.1X-2001 9.4.4, Session Authentication Method,
         802.1X-2004 9.4.4, Session Authentication Method"
    ::= { dot1xAuthSessionStatsEntry 6 }

dot1xAuthSessionTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The duration of the session in seconds."
    REFERENCE
        "802.1X-2001 9.4.4, Session Time,
         802.1X-2004 9.4.4, Session Time"
    ::= { dot1xAuthSessionStatsEntry 7 }

dot1xAuthSessionTerminateCause OBJECT-TYPE
    SYNTAX      INTEGER {
                    supplicantLogoff(1),
                    portFailure(2),
                    supplicantRestart(3),
                    reauthFailed(4),
                    authControlForceUnauth(5),
                    portReInit(6),
                    portAdminDisabled(7),
                    notTerminatedYet(999)
                    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The reason for the session termination."
    REFERENCE
        "802.1X-2001 9.4.4, Session Terminate Cause,
         802.1X-2004 9.4.4, Session Terminate Cause"
    ::= { dot1xAuthSessionStatsEntry 8 }

dot1xAuthSessionUserName OBJECT-TYPE
    SYNTAX     SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The User-Name representing the identity of the
```

```
            Supplicant PAE."
        REFERENCE
            "802.1X-2001 9.4.4, Session User Name,
             802.1X-2004 9.4.4, Session User Name"
        ::= { dot1xAuthSessionStatsEntry 9 }


    -- ---------------------------------------------------------- --
    -- The PAE Supplicant Group
    -- ---------------------------------------------------------- --

    -- ---------------------------------------------------------- --
    -- The Supplicant Configuration Table
    -- ---------------------------------------------------------- --

    dot1xSuppConfigTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Dot1xSuppConfigEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table that contains the configuration objects for the
            Supplicant PAE associated with each port.
            An entry appears in this table for each port that may
            authenticate itself when challenged by a remote system."
        REFERENCE
            "802.1X-2001 9.5.1,
             802.1X-2004 9.5.1"
        ::= { dot1xPaeSupplicant 1 }

    dot1xSuppConfigEntry OBJECT-TYPE
        SYNTAX      Dot1xSuppConfigEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The configuration information for a Supplicant PAE."
        INDEX { dot1xPaePortNumber }
        ::= { dot1xSuppConfigTable 1 }

    Dot1xSuppConfigEntry ::=
        SEQUENCE {
            dot1xSuppPaeState
                INTEGER,
            dot1xSuppHeldPeriod
                Unsigned32,
            dot1xSuppAuthPeriod
                Unsigned32,
            dot1xSuppStartPeriod
                Unsigned32,
            dot1xSuppMaxStart
                Unsigned32,
            dot1xSuppControlledPortStatus
                PaeControlledPortStatus,
            dot1xSuppAccessCtrlWithAuth
                INTEGER,
```



```
        dot1xSuppBackendState
            INTEGER
    }

dot1xSuppPaeState OBJECT-TYPE
    SYNTAX      INTEGER {
                    disconnected(1),
                    logoff(2),
                    connecting(3),
                    authenticating(4),
                    authenticated(5),
                    acquired(6),            -- deprecated
                    held(7),
                    restart(8),
                    sForceAuth(9),
                    sForceUnauth(10)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current state of the Supplicant PAE state
        machine (8.5.8)."
    REFERENCE
        "802.1X-2001 9.5.1, Supplicant PAE State,
         802.1X-2004 9.5.1, Supplicant PAE State"
    ::= { dot1xSuppConfigEntry 1 }

dot1xSuppHeldPeriod OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value, in seconds, of the heldPeriod
        constant currently in use by the Supplicant
        PAE state machine (8.5.8.1.2)."
    REFERENCE
        "802.1X-2001 9.5.1, heldPeriod,
         802.1X-2004 9.5.1, heldPeriod"
    DEFVAL { 60 }
    ::= { dot1xSuppConfigEntry 2 }

dot1xSuppAuthPeriod OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value, in seconds, of the authPeriod
        constant currently in use by the Supplicant
        PAE state machine (8.5.8.1.2)."
    REFERENCE
        "802.1X-2001 9.5.1, authPeriod,
         802.1X-2004 9.5.1, authPeriod"
    DEFVAL { 30 }
    ::= { dot1xSuppConfigEntry 3 }
```

```
dot1xSuppStartPeriod OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value, in seconds, of the startPeriod
        constant currently in use by the Supplicant
        PAE state machine (8.5.8.1.2)."
    REFERENCE
        "802.1X-2001 9.5.1, startPeriod,
         802.1X-2004 9.5.1, startPeriod"
    DEFVAL { 30 }
    ::= { dot1xSuppConfigEntry 4 }

dot1xSuppMaxStart OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of the maxStart constant currently in use by
        the Supplicant PAE state machine (8.5.8.1.2)."
    REFERENCE
        "802.1X-2001 9.5.1, maxStart,
         802.1X-2004 9.5.1, maxStart"
    DEFVAL { 3 }
    ::= { dot1xSuppConfigEntry 5 }

dot1xSuppControlledPortStatus OBJECT-TYPE
    SYNTAX      PaeControlledPortStatus
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current state of the Supplicant PAE state
        machine (8.5.8)."
    REFERENCE
        "802.1X-2001 9.5.1, Supplicant PAE State,
         802.1X-2004 9.5.1, Supplicant PAE State"
    ::= { dot1xSuppConfigEntry 6 }

dot1xSuppAccessCtrlWithAuth OBJECT-TYPE
    SYNTAX      INTEGER {
                    inactive(1),
                    active(2)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The setting for the application of the Supplicant
        authorization state when the port is operating as
        both a Supplicant and an Authenticator.
            inactive  indicates the port will not apply the
                      the Supplicant authorization state, using
                      only the Authenticator authorization
```



```
                        state to restrict access to the port.
                active  indicates the port will apply the
                        the Supplicant authorization state, as
                        well as the Authenticator
                        authorization state."
    REFERENCE
        "802.1X-2004 9.5.1, Supplicant Access Control With
        Authenticator"
    DEFVAL { inactive }
    ::= { dot1xSuppConfigEntry 7 }

dot1xSuppBackendState OBJECT-TYPE
    SYNTAX      INTEGER {
                    initialize(1),
                    idle(2),
                    request(3),
                    response(4),
                    receive(5),
                    fail(6),
                    success(7),
                    timeout(8)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current state of the Supplicant Backend state
     machine."
    REFERENCE
        "802.1X-2004 9.5.1, Backend Supplicant state"
    ::= { dot1xSuppConfigEntry 8 }

-- ------------------------------------------------------------ --
-- The Supplicant Statistics Table
-- ------------------------------------------------------------ --

dot1xSuppStatsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1xSuppStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains the statistics objects for the
        Supplicant PAE associated with each port.
        An entry appears in this table for each port that may
        authenticate itself when challenged by a remote system."
    REFERENCE
        "802.1X-2001 9.5.2,
         802.1X-2004 9.5.2"
    ::= { dot1xPaeSupplicant 2 }

dot1xSuppStatsEntry OBJECT-TYPE
    SYNTAX      Dot1xSuppStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
```

```
        "The statistics information for a Supplicant PAE."
    INDEX { dot1xPaePortNumber }
    ::= { dot1xSuppStatsTable 1 }

Dot1xSuppStatsEntry ::=
    SEQUENCE {
        dot1xSuppEapolFramesRx
            Counter32,
        dot1xSuppEapolFramesTx
            Counter32,
        dot1xSuppEapolStartFramesTx
            Counter32,
        dot1xSuppEapolLogoffFramesTx
            Counter32,
        dot1xSuppEapolRespIdFramesTx
            Counter32,
        dot1xSuppEapolRespFramesTx
            Counter32,
        dot1xSuppEapolReqIdFramesRx
            Counter32,
        dot1xSuppEapolReqFramesRx
            Counter32,
        dot1xSuppInvalidEapolFramesRx
            Counter32,
        dot1xSuppEapLengthErrorFramesRx
            Counter32,
        dot1xSuppLastEapolFrameVersion
            Unsigned32,
        dot1xSuppLastEapolFrameSource
            MacAddress
        }

dot1xSuppEapolFramesRx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL frames of any type
        that have been received by this Supplicant."
    REFERENCE
        "802.1X-2001 9.5.2, EAPOL frames received,
         802.1X-2004 9.5.2, EAPOL frames received"
    ::= { dot1xSuppStatsEntry 1 }

dot1xSuppEapolFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL frames of any type
        that have been transmitted by this Supplicant."
    REFERENCE
        "802.1X-2001 9.5.2, EAPOL frames transmitted,
         802.1X-2004 9.5.2, EAPOL frames transmitted"
```



```
    ::= { dot1xSuppStatsEntry 2 }

dot1xSuppEapolStartFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL Start frames
        that have been transmitted by this Supplicant."
    REFERENCE
        "802.1X-2001 9.5.2, EAPOL Start frames transmitted,
         802.1X-2004 9.5.2, EAPOL Start frames transmitted"
    ::= { dot1xSuppStatsEntry 3 }

dot1xSuppEapolLogoffFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of EAPOL Logoff frames
        that have been transmitted by this Supplicant."
    REFERENCE
        "802.1X-2001 9.5.2, EAPOL Logoff frames transmitted,
         802.1X-2004 9.5.2, EAPOL Logoff frames transmitted"
    ::= { dot1xSuppStatsEntry 4 }

dot1xSuppEapolRespIdFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "The number of EAP Resp/Id frames
        that have been transmitted by this Supplicant."
    REFERENCE
        "802.1X-2001 9.5.2, EAP Resp/Id frames transmitted,
         802.1X-2004 9.5.2, EAP Resp/Id frames transmitted"
    ::= { dot1xSuppStatsEntry 5 }

dot1xSuppEapolRespFramesTx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      deprecated
    DESCRIPTION
        "The number of valid EAP Response frames
        (other than Resp/Id frames)
        that have been transmitted by this Supplicant."
    REFERENCE
        "802.1X-2001 9.5.2, EAP Resp frames transmitted,
         802.1X-2004 9.5.2, EAP Resp frames transmitted"
    ::= { dot1xSuppStatsEntry 6 }

dot1xSuppEapolReqIdFramesRx OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
```

```
        STATUS      deprecated
        DESCRIPTION
            "The number of EAP Req/Id frames
            that have been received by this Supplicant."
        REFERENCE
            "802.1X-2001 9.5.2, EAP Req/Id frames received,
             802.1X-2004 9.5.2, EAP Req/Id frames received"
        ::= { dot1xSuppStatsEntry 7 }

    dot1xSuppEapolReqFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
            "The number of EAP Request frames (other than Rq/Id
            frames) that have been received by this Supplicant."
        REFERENCE
            "802.1X-2001 9.5.2, EAP Req frames received,
             802.1X-2004 9.5.2, EAP Req frames received"
        ::= { dot1xSuppStatsEntry 8 }

    dot1xSuppInvalidEapolFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of EAPOL frames that have been
            received by this Supplicant in which the
            frame type is not recognized."
        REFERENCE
            "802.1X-2001 9.5.2, Invalid EAPOL frames received,
             802.1X-2004 9.5.2, Invalid EAPOL frames received"
        ::= { dot1xSuppStatsEntry 9 }

    dot1xSuppEapLengthErrorFramesRx OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The number of EAPOL frames that have been
            received by this Supplicant in which the Packet
            Body Length field (7.5.5) is invalid."
        REFERENCE
            "802.1X-2001 9.5.2, EAP length error frames received,
             802.1X-2004 9.5.2, EAP length error frames received"
        ::= { dot1xSuppStatsEntry 10 }

    dot1xSuppLastEapolFrameVersion OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The protocol version number carried in the
            most recently received EAPOL frame."
```



```
        REFERENCE
            "802.1X-2001 9.5.2, Last EAPOL frame version,
             802.1X-2004 9.5.2, Last EAPOL frame version"
        ::= { dot1xSuppStatsEntry 11 }

    dot1xSuppLastEapolFrameSource OBJECT-TYPE
        SYNTAX      MacAddress
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The source MAC address carried in the
            most recently received EAPOL frame."
        REFERENCE
            "802.1X-2001 9.5.2, Last EAPOL frame source,
             802.1X-2004 9.5.2, Last EAPOL frame source"
        ::= { dot1xSuppStatsEntry 12 }

    -- ------------------------------------------------------------ --
    -- IEEE 802.1X MIB - Conformance Information
    -- ------------------------------------------------------------ --

    dot1xPaeConformance OBJECT IDENTIFIER ::= { ieee8021paeMIB 2 }

    dot1xPaeGroups OBJECT IDENTIFIER ::= { dot1xPaeConformance 1 }

    dot1xPaeCompliances OBJECT IDENTIFIER
        ::= { dot1xPaeConformance 2 }

    -- ------------------------------------------------------------ --
    -- units of conformance
    -- ------------------------------------------------------------ --

    dot1xPaeSystemGroup OBJECT-GROUP
        OBJECTS {
            dot1xPaeSystemAuthControl,
            dot1xPaePortProtocolVersion,
            dot1xPaePortCapabilities,
            dot1xPaePortInitialize,
            dot1xPaePortReauthenticate
        }
        STATUS      current
        DESCRIPTION
            "A collection of objects providing system information
            about, and control over, a PAE."
        ::= { dot1xPaeGroups 1 }

    dot1xPaeAuthConfigGroup OBJECT-GROUP
        OBJECTS {
            dot1xAuthPaeState,
            dot1xAuthBackendAuthState,
            dot1xAuthAdminControlledDirections,
            dot1xAuthOperControlledDirections,
            dot1xAuthAuthControlledPortStatus,
            dot1xAuthAuthControlledPortControl,
```

```
            dot1xAuthQuietPeriod,
            dot1xAuthTxPeriod,
            dot1xAuthSuppTimeout,
            dot1xAuthServerTimeout,
            dot1xAuthMaxReq,
            dot1xAuthReAuthPeriod,
            dot1xAuthReAuthEnabled,
            dot1xAuthKeyTxEnabled
        }
        STATUS      deprecated
        DESCRIPTION
            "A collection of objects providing configuration
            information about an Authenticator PAE."
        ::= { dot1xPaeGroups 2 }

    dot1xPaeAuthStatsGroup OBJECT-GROUP
        OBJECTS {
            dot1xAuthEapolFramesRx,
            dot1xAuthEapolFramesTx,
            dot1xAuthEapolStartFramesRx,
            dot1xAuthEapolLogoffFramesRx,
            dot1xAuthEapolRespIdFramesRx,
            dot1xAuthEapolRespFramesRx,
            dot1xAuthEapolReqIdFramesTx,
            dot1xAuthEapolReqFramesTx,
            dot1xAuthInvalidEapolFramesRx,
            dot1xAuthEapLengthErrorFramesRx,
            dot1xAuthLastEapolFrameVersion,
            dot1xAuthLastEapolFrameSource
        }
        STATUS      current
        DESCRIPTION
            "A collection of objects providing statistics about an
            Authenticator PAE."
        ::= { dot1xPaeGroups 3 }

    dot1xPaeAuthDiagGroup OBJECT-GROUP
        OBJECTS {
            dot1xAuthEntersConnecting,
            dot1xAuthEapLogoffsWhileConnecting,
            dot1xAuthEntersAuthenticating,
            dot1xAuthAuthSuccessWhileAuthenticating,
            dot1xAuthAuthTimeoutsWhileAuthenticating,
            dot1xAuthAuthFailWhileAuthenticating,
            dot1xAuthAuthReauthsWhileAuthenticating,
            dot1xAuthAuthEapStartsWhileAuthenticating,
            dot1xAuthAuthEapLogoffWhileAuthenticating,
            dot1xAuthAuthReauthsWhileAuthenticated,
            dot1xAuthAuthEapStartsWhileAuthenticated,
            dot1xAuthAuthEapLogoffWhileAuthenticated,
            dot1xAuthBackendResponses,
            dot1xAuthBackendAccessChallenges,
            dot1xAuthBackendOtherRequestsToSupplicant,
            dot1xAuthBackendNonNakResponsesFromSupplicant,
```



```
        dot1xAuthBackendAuthSuccesses,
        dot1xAuthBackendAuthFails
    }
    STATUS      deprecated
    DESCRIPTION
        "A collection of objects providing diagnostic statistics
        about an Authenticator PAE."
    ::= { dot1xPaeGroups 4 }

dot1xPaeAuthSessionStatsGroup OBJECT-GROUP
    OBJECTS {
        dot1xAuthSessionOctetsRx,
        dot1xAuthSessionOctetsTx,
        dot1xAuthSessionFramesRx,
        dot1xAuthSessionFramesTx,
        dot1xAuthSessionId,
        dot1xAuthSessionAuthenticMethod,
        dot1xAuthSessionTime,
        dot1xAuthSessionTerminateCause,
        dot1xAuthSessionUserName
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing statistics about the
        current, or last session for an Authenticator PAE."
    ::= { dot1xPaeGroups 5 }

dot1xPaeSuppConfigGroup OBJECT-GROUP
    OBJECTS {
        dot1xSuppPaeState,
        dot1xSuppHeldPeriod,
        dot1xSuppAuthPeriod,
        dot1xSuppStartPeriod,
        dot1xSuppMaxStart
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing configuration
        information about a Supplicant PAE."
    ::= { dot1xPaeGroups 6 }

dot1xPaeSuppStatsGroup OBJECT-GROUP
    OBJECTS {
        dot1xSuppEapolFramesRx,
        dot1xSuppEapolFramesTx,
        dot1xSuppEapolStartFramesTx,
        dot1xSuppEapolLogoffFramesTx,
        dot1xSuppEapolRespIdFramesTx,
        dot1xSuppEapolRespFramesTx,
        dot1xSuppEapolReqIdFramesRx,
        dot1xSuppEapolReqFramesRx,
        dot1xSuppInvalidEapolFramesRx,
        dot1xSuppEapLengthErrorFramesRx,
        dot1xSuppLastEapolFrameVersion,
```

```
        dot1xSuppLastEapolFrameSource
    }
    STATUS      deprecated
    DESCRIPTION
        "A collection of objects providing statistics about a
        Supplicant PAE."
    ::= { dot1xPaeGroups 7 }

dot1xPaeAuthConfigGroup2 OBJECT-GROUP
    OBJECTS {
        dot1xAuthPaeState,
        dot1xAuthBackendAuthState,
        dot1xAuthAdminControlledDirections,
        dot1xAuthOperControlledDirections,
        dot1xAuthAuthControlledPortStatus,
        dot1xAuthAuthControlledPortControl,
        dot1xAuthQuietPeriod,
        dot1xAuthServerTimeout,
        dot1xAuthReAuthPeriod,
        dot1xAuthReAuthEnabled,
        dot1xAuthKeyTxEnabled
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing configuration
        information about an Authenticator PAE."
    ::= { dot1xPaeGroups 8 }

dot1xPaeSuppConfigGroup2 OBJECT-GROUP
    OBJECTS {
        dot1xSuppControlledPortStatus,
        dot1xSuppAccessCtrlWithAuth,
        dot1xSuppBackendState
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing configuration
        information about a Supplicant PAE."
    ::= { dot1xPaeGroups 9 }

dot1xPaeSuppStatsGroup2 OBJECT-GROUP
    OBJECTS {
        dot1xSuppEapolFramesRx,
        dot1xSuppEapolFramesTx,
        dot1xSuppEapolStartFramesTx,
        dot1xSuppEapolLogoffFramesTx,
        dot1xSuppInvalidEapolFramesRx,
        dot1xSuppEapLengthErrorFramesRx,
        dot1xSuppLastEapolFrameVersion,
        dot1xSuppLastEapolFrameSource
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing statistics about a
```

```
        Supplicant PAE."
    ::= { dot1xPaeGroups 10 }

-- ------------------------------------------------------------ --
-- compliance statements for 802.1X-2001
-- ------------------------------------------------------------ --

dot1xPaeCompliance MODULE-COMPLIANCE
    STATUS  deprecated
    DESCRIPTION
        "The compliance statement for device support of
        Port Access Control."

    MODULE
        MANDATORY-GROUPS {
            dot1xPaeSystemGroup
        }

        GROUP   dot1xPaeAuthConfigGroup
        DESCRIPTION
            "This group is mandatory for systems that support
            the Authenticator functions of the PAE."

        OBJECT  dot1xAuthAdminControlledDirections
        SYNTAX  INTEGER {
                    both(0)
                }
        MIN-ACCESS read-only
        DESCRIPTION
            "Support for in(1) is optional."

        OBJECT  dot1xAuthOperControlledDirections
        SYNTAX  INTEGER {
                    both(0)
                }
        DESCRIPTION
            "Support for in(1) is optional."

        OBJECT dot1xAuthKeyTxEnabled
        MIN-ACCESS read-only
        DESCRIPTION
          "An Authenticator PAE that does not support
          EAPOL-Key frames may implement this object as
          read-only, returning a value of FALSE."

        GROUP   dot1xPaeAuthStatsGroup
        DESCRIPTION
            "This group is mandatory for systems that support
            the Authenticator functions of the PAE."

        GROUP   dot1xPaeAuthDiagGroup
        DESCRIPTION
            "This group is optional for systems that support
            the Authenticator functions of the PAE."
```

```
        GROUP   dot1xPaeAuthSessionStatsGroup
        DESCRIPTION
            "This group is optional for systems that support
            the Authenticator functions of the PAE."

        GROUP   dot1xPaeSuppConfigGroup
        DESCRIPTION
            "This group is mandatory for systems that support
            the Supplicant functions of the PAE."

        GROUP   dot1xPaeSuppStatsGroup
        DESCRIPTION
            "This group is mandatory for systems that support
            the Supplicant functions of the PAE."

    ::= { dot1xPaeCompliances 1 }

-- ---------------------------------------------------------- --
-- compliance statements for 802.1X-2004
-- ---------------------------------------------------------- --

dot1xPaeCompliance2 MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
        "The compliance statement for device support of
        Port Access Control."

    MODULE
        MANDATORY-GROUPS {
            dot1xPaeSystemGroup
        }

        GROUP   dot1xPaeAuthConfigGroup2
        DESCRIPTION
            "This group is mandatory for systems that support
            the Authenticator functions of the PAE."

        OBJECT  dot1xAuthAdminControlledDirections
        SYNTAX  INTEGER {
                    both(0)
                }
        MIN-ACCESS read-only
        DESCRIPTION
            "Support for in(1) is optional."

        OBJECT  dot1xAuthOperControlledDirections
        SYNTAX  INTEGER {
                    both(0)
                }
        DESCRIPTION
            "Support for in(1) is optional."

        OBJECT dot1xAuthKeyTxEnabled
```



```
        MIN-ACCESS read-only
        DESCRIPTION
          "An Authenticator PAE that does not support
          EAPOL-Key frames may implement this object as
          read-only, returning a value of FALSE."

        GROUP   dot1xPaeAuthStatsGroup
        DESCRIPTION
            "This group is mandatory for systems that support
            the Authenticator functions of the PAE."

        GROUP   dot1xPaeAuthSessionStatsGroup
        DESCRIPTION
            "This group is optional for systems that support
            the Authenticator functions of the PAE."

        GROUP   dot1xPaeSuppConfigGroup
        DESCRIPTION
            "This group is mandatory for systems that support
            the Supplicant functions of the PAE."

        GROUP   dot1xPaeSuppStatsGroup2
        DESCRIPTION
            "This group is mandatory for systems that support
            the Supplicant functions of the PAE."

        GROUP   dot1xPaeSuppConfigGroup2
        DESCRIPTION
            "This group is mandatory for systems that support
            the Supplicant functions of the PAE."

    ::= { dot1xPaeCompliances 2 }

END
```

Annex A

(normative)

PICS Proforma[14]

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to this 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 s a statement of the capabilities and options of the protocol that 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, as a detailed indication of the capabilities 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 although interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSs).

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

A.2.1 Status symbols

M Mandatory
O Optional
O.n Optional, but support of at least one of the group of options labeled by the same numeral *n* is required
X Prohibited
pred: Conditional-item symbol, including predicate identification (see A.3.4)
¬ Logical negation, applied to a conditional item's predicate

A.2.2 General abbreviations

N/A Not applicable
PICS Protocol Implementation Conformance Statement

[14]*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.

A.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

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

The main part of the PICS proforma is a fixed-format questionnaire, divided into several subclauses, each containing a number 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. (Note that there are some items in which two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered; the third column records the status of the item—whether support is mandatory, optional, or conditional; see also A.3.4. The fourth column contains the reference or references to the material that specifies the item in the main body of this standard, and the fifth column provides the space for the answers.

A supplier may also provide (or be required to provide) further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled *Ai* or *Xi,* 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 Conformation 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. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing on the answers to some items.

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

A.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 into the Support column, together with an *Xi* reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item.

An implementation for which an Exception item is required in this way does not conform to this standard.

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 status

A.3.4.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply—mandatory 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 selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form "**pred:** S," where **pred** is a predicate as described in A.3.4.2, and S is a status symbol, M or 0.

If the value of the predicate is true (see A.3.4.2), the conditional item is applicable, and its status is indicated by the status symbol following the predicate: the answer column is to be marked in the usual way. If the value of the predicate is false, the "Not Applicable" (N/A) answer is to be marked.

A.3.4.2 Predicates

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma: The value of the predicate is true if the item is marked as supported, and is false otherwise.

b) A predicate-name, for a predicate defined as a boolean expression constructed by combining item-references using the boolean operator OR: The value of the predicate is true if one or more of the items is marked as supported.

c) A predicate-name, for a predicate defined as a boolean expression constructed by combining item-references using the boolean operator AND: The value of the predicate is true if all of the items are marked as supported.

d) The logical negation symbol "¬" prefixed to an item-reference or predicate-name: The value of the predicate is true if the value of the predicate formed by omitting the "¬" symbol is false, and vice versa.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

A.4 PICS proforma for IEEE 802.1X

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) of machines and/or operating system names	
NOTES 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement 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, IEEE Std 802.1X-2004

Identification of protocol specification	IEEE Std 802.1X-2004, IEEE Standards for Local and Metropolitan Area Networks: Standard for Port-based network access control
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	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 IEEE Std 802.1X-2004.)	No [ ] Yes [ ]

Date of Statement	

A.5 Major capabilities and options

Item	Feature	Status	References	Support
*auth	Support the operation of the Port Access Entity (PAE) over the uncontrolled Port, as an Authenticator PAE.	O.1	5.1, 8	Yes [ ] No [ ]
*supp	Support the operation of the Port Access Entity (PAE) over the uncontrolled Port, as a Supplicant PAE.	O.1	5.1, 8	Yes [ ] No [ ]
sysm	Support the system configuration functions.	M	5.1, 9.6.1	Yes [ ]
authM1	Support the ability to configure the operation of the Authenticator.	auth:M	5.1, 9.4.1	Yes [ ] N/A [ ]
authM2	Support the ability to maintain and retrieve Authenticator statistics.	auth:M	5.1, 9.4.2	Yes [ ] N/A [ ]
authM3	Support the operation of the controlled Port consistent with AuthControlledPortControl values of Force Unauthorized, Auto and Force Authorized.	M	5.1, 6.4	Yes [ ] N/A [ ]
authM4	Support the ability to set AuthControlledPort-Control values of Force Unauthorized, Auto and Force Authorized by management.	M	5.1, 6.4, 9.4.1	Yes [ ] N/A [ ]
authM5	Support the operation of the controlled Port consistent with AdminControlledDirections and OperControlledDirections values of Both.	M	5.1, 6.5	Yes [ ] N/A [ ]
authM6	Support regular reauthentication of the Supplicant and configuration of reAuthTimer and reAuthenabled parameters by management.	auth:M	5.1, 8.2.8, 9.4.1	Yes [ ] N/A [ ]
suppM1	Support the ability to configure the operation of the Supplicant.	supp:M	5.1, 9.5.1	Yes [ ] N/A [ ]
suppM2	Support the ability to maintain and retrieve Supplicant statistics.	supp:M	5.1, 9.5.2	Yes [ ] N/A [ ]
bothM1	Support a Supplicant Access Control With Authenticator parameter value of inactive.	auth AND supp: M	5.1, 8.2.2.2, 9.5.1	Yes [ ]
bothO1	Support a Supplicant Access Control With Authenticator parameter values of active and inactive.	auth AND supp: O	5.1, 8.2.2.2, 9.5.1	Yes [ ] No [ ]
other	Support the operation of protocol entities other than the PAE over the uncontrolled Port.	O	5.1	Yes [ ] No [ ]
authO1	Support the ability to maintain and retrieve the Authenticator diagnostics.	auth:O	5.2, 9.4.3	Yes [ ] No [ ]
authO2	Support the ability to maintain and retrieve the Authenticator session statistics.	auth:O	5.2, 9.4.4	Yes [ ] No [ ]
*authO3	Support the operation of the controlled Port consistent with AdminControlledDirections and OperControlledDirections parameter values of Both, and support the ability to set the AdminControlledDirections parameter to the value of Both and In, by management action.	O	5.2, 6.5, 9.4.1	Yes [ ] No [ ]

A.5 Major capabilities and options *(continued)*

Item	Feature	Status	References	Support
*authO4	Support the ability to transmit key information to the Supplicant, and the ability to modify the KeyTransmissionEnabled parameter by management action.	auth:O	5.2, 8.1.9, 8.2.5, 9.4.1	Yes [ ] No [ ]
authO5	Support the ability to transmit and receive key information to and from the Supplicant using alternative but compatible key machines.	auth:O	5.2, 8.1.9, 8.2.5, 8.2.6	Yes [ ] No [ ]
*suppO1	Support the ability to transmit key information to the Authenticator and the ability to modify the KeyTransmissionEnabled parameter by management action.	supp:O	5.2, 8.1.9, 8.2.6, 9.4.1	Yes [ ] No [ ]
*ether	Support EAPOL encapsulation over IEEE 802.3/Ethernet MACs.	O.2	7.2	Yes [ ] No [ ]
*trfddi	Support EAPOL encapsulation over Token Ring/FDDI MACs.	O.2	7.3	Yes [ ] No [ ]
mgt	Support remote management, using the functionality defined in Clause 9, by means of the Port Access Control SNMP MIB.	O	10	Yes [ ] No [ ]

A.6 EAPOL frame formats

Item	Feature	Status	References	Support
eapol	The EAPOL encapsulation used between Authenticator and Supplicant PAEs.	M	7	Yes [ ]
norif	RIF shall not be present in EAPOL frames encapsulated on Token Ring/FDDI.	trfddi:M	7.3	Yes [ ] N/A [ ]
vtag	EAPOL frames shall not be VLAN tagged.	M	7.4	Yes [ ]
ptag1	Support for the reception of priority tagged EAPOL frames.	M	7.4	Yes [ ]
ptag2	Support for the transmission of priority tagged EAPOL frames.	O	7.4	Yes [ ] No [ ]
petype	PAE Ethernet Type field in transmitted EAPOL frames is as defined.	ether:M	7.5.1	Yes [ ] N/A [ ]
psnap	SNAP-encoded Ethernet Type field in transmitted EAPOL frames is as defined.	trfddi:M	7.5.2	Yes [ ] N/A [ ]
pver	Protocol version used in transmitted EAPOL frames is as defined.	M	7.5.3	Yes [ ]
ptype	Reserved values of Packet Type shall not be used in transmitted EAPOL frames.	M	7.5.4	Yes [ ]
pvalid	Frames shall be processed and interpreted according to the validation rules.	M	7.5.7	Yes [ ]
ppvi	Checking of Protocol Version Identifier on receipt.	X	7.5.7	No [ ]

A.6 EAPOL frame formats *(continued)*

Item	Feature	Status	References	Support
padd1	Individual MAC address used as destination address in EAPOL frames, as specified.	M	7.8	Yes [ ]
padd2	Group MAC address used as destination address in EAPOL frames, as specified.	M	7.8	Yes [ ]
padd3	Individual MAC address values used as source address in EAPOL frames.	M	7.8	Yes [ ]

A.7 PAE support

Item	Feature	Status	References	Support
	Relaying EAP and EAPOL frames.			
reap	Relay function does not modify EAP frames.	auth:M	8.1.7	Yes [ ] N/A [ ]
reapol	EAPOL-Start, EAPOL-Logoff, and EAPOL-Key frames are not relayed to EAP.	auth:M	8.1.7	Yes [ ] N/A [ ]
	State machine support.			
mach	The implementation supports the required set of state machines on each Port, in accordance with the PAE role(s) that each Port supports.	M	8.2, Table 8-1	Yes [ ]
timers	The Port Timers state machine is supported as defined.	M	8.2.3, 8-9, 8.2.2.1	Yes [ ]
apsm	The Authenticator PAE state machine is supported as defined.	auth:M	8.2.4, Table 8-10, 8.2.2, 8.2.4.1	Yes [ ] N/A [ ]
akey	The Authenticator Key Transmit state machine is supported as defined.	authO4:M	8.2.5, Table 8-11, 8.2.2, 8.2.5.1	Yes [ ] N/A [ ]
skey	The Supplicant Key Transmit state machine is supported as defined.	suppO1:M	8.2.6, Table 8-12, 8.2.2, 8.2.6.1	Yes [ ] N/A [ ]
rtsm	The Reauthentication Timer state machine is supported as defined.	auth:M	8.2.8, Table 8-14, 8.2.2, 8.2.8.1	Yes [ ] N/A [ ]
basm	The Backend Authentication state machine is supported as defined.	auth:M	8.2.9, Table 8-18, 8.2.2, 8.2.9.1	Yes [ ] N/A [ ]
cdsm	The Controlled Directions state machine is supported as defined.	auth:M	8.2.10, Table 8-16, 8.2.2, 8.2.10.1	Yes [ ] N/A [ ]
cdbd	The Bridge Detection state machine is supported as defined on any Bridge Ports.	bridge:M	8.2.10, Clause 17 of IEEE Std 802.1D	Yes [ ] N/A [ ]
spsm	The Supplicant PAE state machine is supported as defined.	supp:M	8.2.11, Table 8-17, 8.2.2, 8.2.11.1	Yes [ ] N/A [ ]

A.7 PAE support *(continued)*

Item	Feature	Status	References	Support
spbe	The Supplicant Backend state machine is supported as defined.	supp:M	8.2.12, 8-18, 8.2.2, 8.2.12.1	Yes [ ] N/A [ ]
skey	The Key Receive state machine is supported as defined.	M	8.2.7, 8-13, 8.2.2, 8.2.7.1	Yes [ ] N/A [ ]

PREDICATES:
bridge = auth AND auth03 AND {the Port is a Bridge Port}

Annex B

(informative)

Scenarios for the use of Port-Based Network Access Control

B.1 Rationale for unidirectional control functionality

The ability to set the AdminControlledDirections parameter for a Port to In (see 6.5) has been included in this standard primarily with the intent of supporting a number of features of the PC environment, including

a) Remote wakeup

b) Peer-to-peer wakeup

c) Remote control

d) Alerting

B.1.1 Remote wakeup

Remote wakeup allows a management console that is connected within the LAN (or that can establish a connection to the LAN via a WAN connection) to perform maintenance functions on PCs that are connected to the periphery of the LAN, but may or may not be powered on at the time. The management console transmits a “Magic Packet” to the PC; the LAN adaptor recognizes the Magic Packet and causes the PC to be activated.

In an IEEE 802.1X environment that is configured for Full Control, powering down the PC will result in loss of authentication, and the Magic Packet will be blocked by the Bridge Port to which it is connected, effectively disabling the remote wakeup capability.

The use of unidirectional control allows these Magic Packets to be relayed through the Bridge Port, restoring the remote wakeup capability. However, as unidirectional control will allow any frames (not just Magic Packets) to leak out of the Port, the use of this feature inevitably results in a reduction of the protection afforded by the use of Port Access Control.

B.1.2 Peer-to-peer wakeup

In PC environments that employ Windows peer-to-peer networking, any Windows client can share its resources with other PCs attached to the LAN. As with remote wakeup, if a PC that has offered its resources on the network is power managed (for example, in a standby state), then those shared resources are not available to other clients on the LAN.

Windows provides the MAC driver of the PC with a number of frame patterns that, if received by the MAC, cause it to invoke the relevant restore function, in a similar manner to remote wakeup. However, the Controlled Port of the Bridge to which the PC is attached may have transitioned to Disabled, and if Full Control is in force, these frames will be blocked by the Bridge Port that connects the PC to the LAN. Again, the use of unidirectional control allows the wakeup function to be used, but with consequent lowering of the level of protection offered.

B.1.3 Remote control

The Alerting Standards Forum (ASF, part of the Desktop Management Forum) is proposing standards for remote control (power on/off, reboot, etc.) of clients that support authentication. The functions proposed use a similar approach to the Remote Wakeup "Magic Packet"; as with the previous examples, it relies on the ability to transmit a packet out of the Bridge Port to the client system in order to activate the function. Again, the use of unidirectional control offers a means of allowing this functionality to be implemented.

B.1.4 Alerting

Within ASF (and other fora), specifications of alert messages are being developed that allow a PC or workstation to signal to management that some problem exists with its physical or operational environment, for example, overheating, fan stopped, power supply problems, and so on. These alerts would generally take the form of SNMP trap packets. If the switch Port that serves that end station supports Port Access Control, then these alerts will be blocked by the Controlled Port if it is in the Unauthorized state, regardless of the Control Mode setting for the Port.

In order to allow these alerts to be forwarded to a management station within the LAN, it is necessary to make use of the Uncontrolled Port to receive and process such alerts. An alert handler attached to the Uncontrolled Port can recognize those incoming frames that contain validly specified alert messages, and selectively forward them to management stations within the LAN.

B.1.4.1 Alert encapsulation

The Packet Type of EAPOL-Encapsulated-ASF-Alert (see 7.5.4) provides a means in which alert messages can be encapsulated within an EAPOL frame and can be recognized as such by the protocol entities attached to the Uncontrolled Port.

B.1.4.2 Alert generation by the Supplicant system

The station that generates the ASF alert has two possible strategies available to it:

a) Transmit the alert in two forms at all times: one encapsulated in an EAPOL frame, and one in the native form for the alert concerned (e.g., an SNMP Trap).

b) Use its knowledge of whether it has successfully completed authentication with an Authenticator to determine whether to send a native alert or both native and encapsulated alerts. If the Supplicant has received a positive response to its authentication exchanges (i.e., received EAP-Success), then the Authenticator's controlled Port can be assumed to be in the Authorized state, and therefore, there is no need to send the encapsulated alert. However, if the Supplicant has simply timed out the Authenticator, or has received EAP-Fail, then both versions of the alert are sent, as the Supplicant cannot rely on the Controlled Port being in the Authorized state.

Either of these approaches is acceptable.

B.1.4.3 Alert handling by the Authenticator system

When an EAPOL frame with this Packet Type is received on an Uncontrolled Port, the receiving Authenticator PAE passes the frame to the protocol entity responsible for handling ASF alerts for further processing. It is assumed that the specification of the ASF protocol entity will clearly determine the format of an acceptable ASF alert message, and that any EAPOL-Encapsulated-ASF-Alert frame that carries a Packet Body that does not meet these criteria will be discarded by the ASF protocol entity.

The ASF protocol entity can adopt either of the following strategies with regard to received EAPOL-Encapsulated-ASF-Alert frames that meet its acceptance criteria:

a) Make use of local knowledge of the state of the controlled Port to determine whether the EAPOL-Encapsulated-ASF-Alert frame should be decapsulated and forwarded on, or discarded. If the controlled Port is in the Authorized state, the ASF protocol entity discards any EAPOL-Encapsulated-ASF-Alert frames, as the Supplicant will send an unencapsulated copy of the same alert.

b) Ignore the state of the controlled Port, and decapsulate and forward on the alert.

It should be noted that when the controlled Port is in the Authorized state, any incoming EAPOL-Encapsulated-ASF-Alert frames are received on both the controlled and uncontrolled Ports. If the Port is a Bridge Port, then the presence of the Port Access Entity Ethernet Type as the destination MAC address ensures that the relay function of the Bridge cannot relay such frames to its outbound Ports.

NOTE—The expected maximum rate of alerts that will be handled using this mechanism is less than 10 alerts per second.

B.1.4.4 Implications for the recipient of ASF alerts

The combination of the Supplicant and Authenticator strategies for handling EAPOL-Encapsulated-ASF-Alert frames means that the eventual recipient of the ASF alert messages can receive two copies of the same alert message.

B.2 Use of IEEE 802.1X in point-to-point and shared media LANs

The original intent behind the development of IEEE Std 802.1X was to leverage the characteristics of point-to-point, dedicated physical connections in LANs, as exist, for example, in switched LAN infrastructures. As any single LAN segment in such infrastructures has no more than two devices attached to it, one of which is a Bridge Port, these infrastructures provide a useful opportunity to develop simple access control capability. The Bridge Port can monitor the operational state of its MAC, allowing it to detect events that indicate the attachment of an active device at the remote end of the link, or an active device becoming inactive; these events can be used to control the authorization state of the Port and to initiate the process of authenticating the attached device if the Port is unauthorized.

This assumption, that the Port is dealing with a point-to-point connection to a single connected device, is a key assumption in terms of the security offered by Port Access Control; once the connected device has successfully been authenticated, then the Port becomes Authorized, and all subsequent traffic on the Port is not subject to access control restriction until an event occurs that causes the Port to become Unauthorized. Hence, if the Port is actually connected to a shared media LAN segment with more than one attached device, successfully authenticating one of the attached devices effectively provides access to the LAN for all devices on the shared segment. Clearly, the security offered in this situation is not terribly high and is open to attack.

In order to successfully make use of IEEE Std 802.1X in a shared media LAN segment, it would be necessary to create “logical” Ports, one for each attached device that required access to the LAN, and to ensure that traffic carried by these Ports is secure by applying encryption not only to the data traffic on the Port, but also to the EAPOL exchanges. The Bridge would, in this case, regard the single physical Port connecting it to the shared media segment as consisting of a number of distinct logical Ports, each logical Port being independently controlled from the point of view of EAPOL exchanges and authorization state, and each carrying encrypted data and control frames.

A special case of shared media access exists in IEEE 802.11 Wireless LANs, in which a station must form an association with an access point in order to make use of the LAN. The protocol that establishes the association allows the station and access point to learn each others' individual MAC addresses, and effectively creates a logical Port that the station can then use to communicate with the LAN via the access point. Once the association has been established, the association allows IEEE 802.1X Port access control to authenticate the attached station, and for the access point to authorize the logical Port.

The IEEE 802.1X specification includes a number of features aimed specifically at supporting the use of Port Access Control in IEEE 802.11 LANs, as secure access is an important requirement for the successful deployment of wireless LAN technology on a large scale. These features may also prove useful in deploying IEEE Std 802.1X in other shared media environments:

a) The ability to make use of the individual MAC addresses of the station and access point as the destination address in EAPOL protocol exchanges.

b) The ability for the access point to distribute or obtain global key information to/from attached stations, by means of the EAPOL-Key message, following successful authentication.

Annex C

(informative)

Design considerations and background material for Port-Based Network Access Control

C.1 Design considerations

C.1.1 Edge authentication in a Bridged Local Area Network

Where Port-based access control is used in MAC Bridges, authentication occurs at the first point of attachment to the Bridged Local Area Network (i.e., at the local access Bridge). Authentication is initiated by the PAE on a Port of the Bridge when

a) A change in the state of a Port from disabled to enabled indicates that there may now be a Supplicant device that is accessible on that Port. Until authentication takes place, the Port is assumed to be unauthorized.

b) The PAE determines that there is a need to reconfirm the authorization state of a Port as a result of a timeout expiry.

c) The Port is unauthorized, but a Supplicant device is attempting to transmit data frames on the LAN segment.

d) The Supplicant device makes an explicit request to initiate the authentication process.

Requiring that authentication occur at the edge rather than in the core of the LAN has several advantages, as follows:

e) **Security.** All authenticated end stations on the local access Bridge are protected from nonauthenticated end stations. If authentication was performed on a core Bridge, it would be possible for a malicious end station to attack authenticated end stations connected to the same local access Bridge or any number of other local access Bridges between this Bridge and the core Bridge. These attacks are eliminated by limiting service to nonauthenticated end stations directly on the local access Bridge.

f) **Complexity.** If authentication is performed in the core of the LAN, there would be the possibility of multiple Bridges on a shared segment initiating authentication. To avoid this, the Bridge Protocol Entity would have to manipulate the Spanning Tree states to make sure that only the Bridge that lies in the forwarding path initiates authentication.

g) **Scalability.** Implementing authentication in the core of the LAN would require authentication to depend on individual MAC addresses, not just on physical point of attachment. This in turn would require that the authentication state be associated with the Filtering Database entry for that MAC address. This increases the implementation cost and would require changes to the operation of address aging and learning. Topology changes and spanning tree reconfiguration complicate the interaction in a large network.

h) **Availability.** Bridged networks are frequently designed with availability as one of the primary goals. The core of the network is redundant and fault-tolerant. If authentication is performed in the core, it would require reauthentication of all the end stations whenever topology changes cause Port state changes in the spanning tree.

i) **Translational Bridging.** Performing authentication at the access Bridge avoids complications arising from translational Bridging or VLANS. If only a single link exists between the end station and the Bridge, frames need not be translated or tagged during the authentication exchange. The path to a core Bridge may involve a variety of link types (FDDI, Token-Ring, etc.) and packet formats (e.g., VLAN tagged frames, MAC encapsulations). Were authentication to be allowed on core Bridges, additional rules may be necessary in order to specify how the authentication protocols are translated.

j) **Multicast propagation.** Were authentication to occur in the core of the LAN, it would be necessary for local access Bridges to forward authentication traffic toward the core so that the Authenticator could respond. Because core Bridges are not able to sense end-station connection to the local access Bridge Port, initiation of authentication would occur either on receiving traffic from a new end station or via end-station initiation. Requiring a core Bridge to maintain authentication state for each end station does not scale. In order for end-station initiation to reach the core Bridge, this would require that these (multicast) frames be flooded by authentication-unaware access Bridges, impacting other end stations. In contrast, if authentication occurs only on local access Bridges, these multicast frames are not forwarded.

C.1.2 Use with IEEE 802.3 Link Aggregation

The requirement stated in 6.8, namely, that Port Access Control acts on physical Ports and not aggregated Ports, is a consequence of the fundamental structure of Port Access Control and its use of addressing.

The model of Port Access Control is that a single Authenticator establishes a dialogue with a single Supplicant in order to establish its credentials. On a point-to-point link, the nature of the connection technology ensures that there can only be two devices attached to the link, and so group MAC addresses can be used to establish the conversation, avoiding the need for the parties to discover their respective, individual MAC addresses prior to communication taking place. The use of group addressing also allows EAPOL to take advantage of the characteristics of the set of reserved group MAC addresses that are defined not to be propagated by MAC Bridges, avoiding the problems that might occur should EAPOL frames be allowed to "leak" through Bridges onto other LAN links.

An exception to this use of group addressing is found in the use of Port Access Control with IEEE 802.11; however, in this case, the operation of IEEE 802.11 ensures that an association exists between the station and the access point, and individual MAC addresses exchanged, before any EAPOL exchanges take place. EAPOL exchanges can then make use of the logical Port created by the association.

A consequence of the use of this model is that it is not possible to "stack" multiple instances of Port Access control on top of each other in a hierarchical structure, and where it might be considered possible to do this (for example, where a mechanism like IEEE 802.3 Link Aggregation allows logical Ports to be formed from one or more physical Port), a choice has to be made as to the level in the hierarchy of logical and physical ports where the access control mechanism will be applied. An example will illustrate why this is the case.

In Figure C-1, two systems are illustrated; System S is a Supplicant system, and System A is an Authenticator system. Each system has two MACs, connected to two point-to-point LAN segments; MAC 1 in each system connects to LAN 1, and MAC 2 to LAN 2. Both systems implement IEEE 802.3 Link Aggregation; so, ignoring the operation of Port Access Control for the time being, the natural final configuration for these two systems is that MACs 1 and 2 would be aggregated to Aggregator 1 in both systems, provided that both systems have configured these links to be aggregatable and to have the same local Key values. Hence, Aggregator 2 in each system would end up with no MACs connected to it, and Aggregator 1 would provide the only operable Port above the Link Aggregation sublayer.

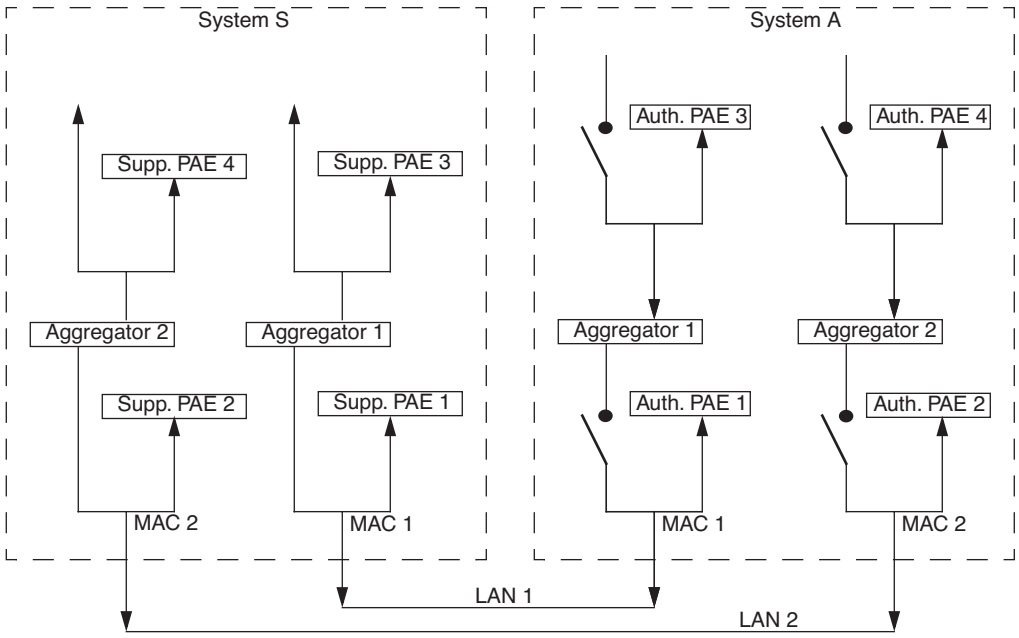


Figure C-1—The problem with Port Access Control in hierarchical structures

Clearly, the potential exists to create controlled and uncontrolled Ports as shown for System A, with Authenticator PAEs associated with each controlled/uncontrolled Port pair. Similarly, in System S, Supplicant PAEs could be attached in the positions shown, above or below the Aggregators. However, if you put Port Access Control in both places, as shown in Figure C-1, the fact that communication between the two systems relies on the use of a single destination address (the EAPOL group address, or the MAC address of the destination Port) means that the receiving system cannot tell which level in the hierarchy should handle a given incoming message. Hence, it is essential that both systems make a choice of where Port Access Control will reside. There are essentially three possible approaches that could be taken here:

a) If the choice is made to place Port Access Control below the Aggregators in both systems, then it will be necessary to authorize both Ports before aggregation can take place. This can be achieved by ensuring that the participating systems regard all Ports as nonaggregatable while they are in the Unauthorized state; a port therefore cannot join an aggregate of more than one link until it becomes Authorized, and it is forced to leave an aggregate of more than one link if it subsequently becomes Unauthorized. Placing the controlled Ports at this level in the structure also forces any Aggregator that is supported by an unauthorized Port to be inoperable. At the Aggregator level, you can therefore be certain that any Aggregator that is operable is supported by one or more links that have been authorized.

b) If the choice is made to place Port Access Control above the Aggregators in both systems, then it is possible for aggregation to take place regardless of the authorization state. The absence of any authentication associated with the individual physical Ports opens up the possibility of attack. For example, an attack could be simply and successfully launched by a third system masquerading as System S or A (in this case, all it would have to do to achieve this would be to borrow the other system's MAC address); a link could then be included in an aggregation by one or another system, despite it being attached to the third system, and the application of Port Access Control above the Aggregator would be incapable of detecting that this had happened.

c) If it is left up to the implementor to decide where the access control mechanism is applied, then this merely combines the addressing ambiguities caused by putting the mechanism in both places with the inherent insecurity of placing Port Access Control above the Aggregators without first authenticating the individual physical Ports.

From the previous discussion, it is clear that the only workable solution is to perform authentication on a per-physical Port basis, and to make this a conformance requirement in order to ensure that implementations that deploy IEEE 802.1X and IEEE 802.3 Link Aggregation in the same system will be interoperable, hence, the conformance requirements stated in 6.8. Clearly, should further configurations be identified in which a choice needs to be made as to where Port access Control should be positioned, then further conformance requirements may be needed in order to ensure correct operation and interoperability.

As mentioned briefly in 6.8, the fact that Ports that are candidates for aggregation are required to be authorized before the aggregation takes place opens up the potential threat of a Port, authorized to entity A, joining an existing aggregation with a second Port, authorized to entity B, and stealing some of the aggregation's traffic. Hence, secure operation of Link Aggregation depends upon either:

a) Ensuring that all Ports that are candidates to join a given aggregation are authorized to the same entity; or

b) Ensuring that all Ports that are candidates to join a given aggregation are authorized to entities that are entitled to belong in the same aggregation.

This can be achieved by manipulating the aggregation keys associated with each port depending upon the outcome of the authentication; for example, by making use of parameters returned by RADIUS with an Access-Accept message.

C.2 Additional services

This subclause describes additional services that can be provided along with Port-based access control. Some of these services are enabled by IEEE 802.1Q VLANs. This is done to give the reader some insight into different operating environments and not meant to be an addition to the specification contained in this standard.

C.2.1 Manageability of end stations

In many installations, it is essential that network management traffic be allowed between the network management station and end stations, in order to permit activities such as network monitoring and software update. If many end stations were to be made inaccessible as a result of failed authentications, network management capabilities would be compromised.

For example, as a result of a power failure, it is conceivable that many end stations would be unable to successfully authenticate, and as a result might be unable to locate a DHCP server. Were these end stations to be completely cut off from the network, then they would never receive a routable IP address, and network administrators would be unable to diagnose the problem, because the end stations would not be reachable by the network management station.

To address this issue, authentication-aware Bridges may support VLAN policy. This allows the Bridge to assign a VLAN to a Port based on the outcome of authentication. In authentication-aware Bridges supporting VLANs, a Port is put into the "forwarding" state during authentication, permitting access to the "nonauthenticated" VLAN. Once authentication has succeeded, a new VLAN ID is assigned for that Port, and the Port remains in "forwarding" state.

Bridge support for a nonauthenticated VLAN enables end stations failing authentication to obtain IP addresses via DHCP so that they can remain manageable. This is useful for enabling end stations to obtain an account and login credentials via a registration server. This also makes it possible to keep track of unauthenticated end stations and manage them if necessary.

C.2.2 Accounting and policies

Authentication-enabled Bridges may support additional services such as accounting or QoS policy. For example, after a connection is sensed on a Port, a timer can be reset, or after authentication succeeds, the Bridge can reset the Port counters. This allows the Bridge to keep track of how long connectivity was maintained on a Port or how many octets were sent in and out. It is also possible for the Bridge to tag packets entering or leaving the Port with a given priority, based on the end-station identity.

C.2.3 End-station identity for access

The identity presented by the end station may either correspond to a user, group, or machine identity. End-station implementations supporting use of a machine identity will typically authenticate once at startup, and will remain authenticated until a "Port down" event occurs or the end station or Bridge reinitialization or reauthentication occurs. In such implementations, accounting data will indicate a single long-lived session for end station. Thus, it will not be possible to account for usage by user. In contrast, implementations supporting user or group identity may authenticate with each user login. In such implementations, accounting data will provide per-user information.

C.2.4 VLAN enhancements

As discussed previously, VLANs can be used to facilitate management of end stations failing authentication.

When the EAPOL-Logoff mechanism is used with VLAN-enabled Bridges, the end station is placed in a "nonauthenticated" VLAN during the period between logins. In order to maintain IP connectivity, the end station would need to release its DHCP address, and acquire a new address so that it would be functional in the nonauthenticated VLAN during the period between logins.

Note that with a VLAN-enabled Bridge, Supplicant initiation is required in order to guarantee assignment of an authenticated address. When VLANs are supported, a DHCP server will typically be provided on the unauthenticated VLAN. As a result, without end-station initiation, the initial DHCP packet sent by the end station could reach the DHCP server on the unauthenticated VLAN, allowing the DHCP conversation to complete prior to authentication. The result is that an authentication-capable end station will be assigned to the nonauthenticated VLAN. Timing problems are less likely for Bridges without VLAN support, because the Port will be blocked, and thus, the DHCP conversation cannot complete prior to authentication.

C.3 Security considerations

The following security issues have been identified as relating to Bridge Port authentication:

a) Piggybacking

b) Snooping

c) Crosstalk

d) Rogue Bridge

e) Bit flipping

f) Negotiation attacks

These issues are discussed in the following subclauses.

C.3.1 Piggybacking

Because it is possible that more than one end station may be connected to a Bridge Port, a Bridge implementing this specification may support anti-piggybacking functionality. Piggybacking occurs when an unauthenticated end station gains access to the Bridge Port based on the successful authentication of another end station. In order to enable piggybacking prevention, authentication-aware Bridges must be configurable on a per-Port basis to set an alarm or block Port access when multiple end stations are detected. Techniques that provide data confidentiality can also be used to block piggybacking.

NOTE—This consideration is intended to deal with situations in which PACP is deployed in switched LANs, but in which nonstandard devices, such as "buffered repeaters," provide shared access to a switched Port. Where PACP is supported in shared media environments that provide some means of establishing an association between the two parties, for example, when used in IEEE 802.11 networks, it may be inappropriate to block access on detection of multiple end stations.

C.3.2 Snooping

In this attack, an attacker on the same Bridge Port listens in on the authentication conversation in an effort to gain further information useful in an attack. Because EAP transmits the Identity in the clear, it is possible for an attacker to learn the identity of users authenticating to the Bridge. However, password compromise can be avoided by use of EAP methods employing strong cryptography.

C.3.3 Crosstalk

In this attack, a Supplicant on one Port attempts to interfere with authentications occurring on another Port. For example, a Supplicant may send an EAP-Failure message to the broadcast address, or to a Supplicant on another Port, or it may send an EAP-Response to the MAC address of another Bridge Port.

In order to prevent crosstalk between Ports attached to wired (as opposed to wireless) LANs, authentication-enabled Bridges must discard all frames with the PAE Ethernet Type and a destination address other than the Bridge Port MAC address or the multicast address. In addition, authentication-enabled Bridges must not leak EAP frames destined for the multicast address to other Ports.

Alternatively, an end station on another LAN may attempt to send packets that will interfere with Bridge Port authentication occurring on another segment. However, this is not possible because EAPOL frames are not routable.

C.3.4 Rogue Bridge

In this attack, the attacker replaces the Bridge with a suitably modified device. In such an attack, the attacker could send an EAP-Request with a lesser form of authentication (for example, EAP-MD5 with a static challenge) in order to perpetrate a dictionary attack and recover the user's password. This attack can be prevented by configuring the client to require an EAP-type supporting mutual authentication. Alternatively, if a mutually authenticating method were used, such as EAP TLS (see IETF RFC2716), the Rogue Bridge could send a premature EAP Success, fooling the Supplicant into terminating authentication prior to authenticating itself to the Supplicant. This attack can be prevented by having the EAP method indicate whether it has completed mutual authentication, and having the Supplicant refuse to process an EAP Success received prior to that indication.

C.3.5 Bit flipping

The goal of EAP is to provide extensible authentication. Other security services, including integrity protection, encryption, or replay protection, are not provided by this proposal. If such services are desired, then it is recommended that other solutions that provide security associations, such as IPsec, be employed.

C.3.6 Negotiation attacks

In this attack, the attacker attempts to subvert the EAP negotiation by inserting or modifying packets on the wire. The goal of this attack is to deny service or to reduce the level of security negotiated between the Bridge and the Supplicant.

While individual EAP authentication types may provide message integrity protection for the data portion of EAP-Request and EAP-Response packets, the EAP header is not integrity protected. In addition, EAP-Success and EAP-Failure messages are not integrity protected, nor are EAP-Request and EAP-Response packets of types Identity, NAK, OTP, or MD-5.

This means that an attacker can send an EAP-Failure message to the Supplicant from the Bridge's MAC address without fear of detection. Also, in response to an EAP-Request sent by the Bridge, the attacker could send an EAP-NAK in an attempt to cause the Bridge and Supplicant to negotiate down to a less secure form of authentication.

While such attacks can result in a denial of service, the attacker must have physical access to the Bridge Port in order to carry them out. Such attacks are detectable by Bridges, RMON probes, or sniffers, and they can be made more difficult by having Bridges employ spoofing protection, i.e., dropping incoming frames claiming to originate from Bridge MAC addresses.

Subversion of the authentication negotiation can be averted using negotiation policy on the Supplicant and Bridge. For example, the Supplicant or Bridge can be configured to only accept a single form of authentication for a claimed Identity.

Integrity protection of EAP messages, including types of Identity, NAK, Notification, Success and Failure is supported by EAP methods that set up a protected channel, conduct the EAP conversation within that channel, and then cryptographically bind the protected channel to the EAP conversation. In situations where there is a substantial risk of spoofing, use of one of these methods may be advisable.

Annex D

(informative)

IEEE 802.1X RADIUS Usage Guidelines[15]

D.1 Introduction

IEEE 802.1X provides "network port authentication" for IEEE 802 media, including Ethernet, Token Ring, and IEEE 802.11 wireless LANS.

IEEE 802.1X does not require use of a central Authentication Server, and thus can be deployed with stand-alone Bridges or access points, as well as in centrally managed scenarios.

In situations where it is desirable to centrally manage authentication, authorization, and accounting (AAA) for IEEE 802 networks, deployment of a central authentication and accounting server is desirable. In such situations, it is expected that IEEE 802.1X Authenticators will function as AAA clients.

D.2 RADIUS accounting attributes

With a few exceptions, the RADIUS accounting attributes defined in IETF RFC 2866 and IETF RFC 2869 have the same meaning within IEEE 802.1X sessions as they do in dialup sessions, and therefore, no additional commentary is needed. Attributes requiring more discussion include the following:

a) Acct-Terminate-Cause

b) Acct-Multi-Session-Id

c) Acct-Link-Count

d) Port-Limit

D.2.1 Acct-Terminate-Cause

This attribute indicates how the session was terminated, as described in IETF RFC 2866. The set of Session Terminate Cause values identified in 9.4.4.1.3, and their mapping onto the RADIUS Acct-Terminate-Cause values, are shown in Table D-1.

When using this attribute, the User Request (1) termination cause corresponds to the situation in which the session terminated due to an EAPOL-Logoff received from the Supplicant. When a session is moved due to roaming, the EAPOL state machines will treat this as a Supplicant Logoff.

A Lost Carrier (2) termination cause indicates session termination due to loss of physical connectivity for reasons other than roaming (see IETF RFC 2607). For example, if the Supplicant disconnects a point-to-point LAN connection, or moves out of range of an IEEE 802.11 Access Point, this termination cause is used. Lost Carrier (2) therefore equates to a Port Disabled condition in the EAPOL state machines.

A Supplicant Restart (19) termination cause indicates reinitialization of the Supplicant state machines.

[15]The material in this annex was derived from IETF RFC 3580, developed in collaboration between participants in the Internet Engineering Task Force (IETF) and the IEEE 802.1 Working Group.

Table D-1—Mappings onto Acct-Terminate-Cause

Session Terminate Cause value	Acct-Terminate-Cause value
supplicantLogoff (1)	User Request (1)
portFailure (2)	Lost Carrier (2)
supplicantRestart (3)	Supplicant Restart (19)
reauthFailed(4)	Reauthentication Failure (20)
authControlForceUnauth(5)	Admin Reset (6)
portReInit(6)	Port Reinitialized (21)
portAdminDisabled (7)	Port Administratively Disabled (22)
notTerminatedYet (999)	N/A

A Reauthentication Failure (20) termination cause indicates that a previously authenticated Supplicant has failed to reauthenticate successfully following expiry of the reauthentication timer or explicit reauthentication request by management action.

Within IEEE Std 802.11, periodic reauthentication may be useful in preventing reuse of an initialization vector with a given key. Since successful reauthentication does not result in termination of the session, accounting packets are not sent as a result of reauthentication unless the status of the session changes. For example:

a) The session is terminated due to reauthentication failure. In this case the Reauthentication Failure (20) termination cause is used.

b) The authorizations are changed as a result of a successful reauthentication. In this case, the Service Unavailable (15) termination cause is used. For accounting purposes, the portion of the session after the authorization change is treated as a separate session.

Where IEEE 802.1X authentication occurs prior to IEEE 802.11 association, accounting packets are not sent until an association occurs.

An Admin Reset (6) termination cause indicates that the Port has been administratively forced into the unauthorized state.

A Port Reinitialized (21) terminate cause indicates that the Port's MAC has been reinitialized.

A Port Administratively Disabled (22) terminate cause indicates that the Port has been administratively disabled.

D.2.2 Acct-Multi-Session-Id

The purpose of this attribute is to make it possible to link together multiple related sessions. For example, it is possible for a Supplicant roaming between IEEE 802.11 Access Points to cause multiple RADIUS accounting packets to be sent by different Access Points.

Where supported by the Access Points, the Acct-Multi-Session-Id attribute is used to link together the multiple related sessions of a roaming Supplicant. In such a situation, if the session context is transferred between

access points, accounting packets may be sent without a corresponding authentication and authorization exchange. However, in such a situation it is assumed that the Acct-Multi-Session-Id is transferred between the Access Points as part of the Inter-Access Point Protocol.

If Acct-Multi-Session-Id was not unique between IEEE 802.11 Access Points, then it is possible that the chosen Acct-Multi-Session-Id may overlap with an existing value allocated on that Access Point, and the Accounting Server would therefore be unable to distinguish a roaming session from a multilink session.

As a result, the Acct-Multi-Session-Id attribute is unique among all the Access Points, Supplicants, and sessions. In order to provide this uniqueness, it is suggested that the Acct-Multi-Session-Id be of the form:

Original Access-Point MAC Address | Supplicant MAC Address | NTP Timestamp

Here "|" represents concatenation, the original Access-Point MAC Address is the MAC address of the Access Point at which the session started, and the 64-bit NTP timestamp indicates the beginning of the original session. In order to provide for consistency of the Acct-Multi-Session-Id between roaming sessions, the Acct-Multi-Session-Id may be moved between Access Points as part of an interaccess point protocol or another handoff scheme.

The use of Acct-Multi-Session-Id of this form guarantees uniqueness among all Access Points, Supplicants, and sessions. Because the NTP timestamp does not wrap on reboot, there is no possibility that a rebooted Access Point could choose an Acct-Multi-Session-Id that could be confused with that of a previous session.

Since the Acct-Multi-Session-Id is of type String as defined in IETF RFC 2866, for use with IEEE 802.1X, it is encoded as an ASCII string of Hex digits. Example: "00-10-A4-23-19-C0-00-12-B2-14-23-DE-AF-23-83-C0-76-B8-44-E8"

D.2.3 Acct-Link-Count

The Acct-Link-Count attribute may be used to account for the number of ports that have been aggregated. If the attribute is not sent, the assumption is that the port is not aggregated.

D.3 RADIUS authentication

This subclause describes how attributes defined in IETF RFC 2865, IETF RFC 2867, IETF RFC 2868, IETF RFC 3579, and IETF RFC 3162 are used in IEEE 802.1X authentication.

D.3.1 User-Name

In IEEE Std 802.1X, the Supplicant typically provides its identity via an EAP-Response/Identity message. Where available, the Supplicant identity is included in the User-Name attribute and included in the RADIUS Access-Request and Access-Reply messages as specified in IETF RFC 2865 and IETF RFC 3579.

Alternatively, as discussed in IETF RFC 3579, Section 2.1, the User-Name attribute may contain the Calling-Station-ID value, which is set to the Supplicant MAC address.

D.3.2 User-Password, CHAP-Password, CHAP-Challenge

Since IEEE Std 802.1X does not support PAP or CHAP authentication, the User-Password, CHAP-Password, or CHAP-Challenge attributes are not used by IEEE 802.1X Authenticators acting as RADIUS clients.

D.3.3 NAS-IP-Address

For use with IEEE Std 802.1X, the NAS-IP-Address contains the IPv4 address of the Bridge or Access Point acting as an Authenticator, and the NAS-IPv6- Address contains the IPv6 address. If the Authenticator has more than one interface, it may be desirable to use a loopback address for this purpose so that the Authenticator will still be reachable even if one of the interfaces was to fail.

D.3.4 NAS-Port

For use with IEEE Std 802.1X, the NAS-Port will contain the port number of the bridge, if this is available. While an Access Point does not have physical ports, a unique "association ID" is assigned to every mobile Station upon a successful association exchange. As a result, for an Access Point, if the association exchange has been completed prior to authentication, the NAS-Port attribute will contain the association ID (a 16-bit unsigned integer), in the lower 16 bits of the 32-bit NAS-Port attribute. Where IEEE 802.1X authentication occurs prior to association, a unique NAS-Port value may not be available.

D.3.5 Service-Type

For use with IEEE Std 802.1X, the Framed (2), Authenticate Only (8), and Call Check (10) values are most commonly used:

a) A Service-Type of Framed (2) indicates that appropriate IEEE 802 framing should be used for the connection.

b) A Service-Type of Authenticate Only (8) indicates that no authorization information needs to be returned in the Access-Accept.

c) As described in IETF RFC 2865, a Service-Type of Call Check (10) is included in an Access-Request packet to request that the RADIUS server accept or reject the connection attempt, typically based on the Called-Station-ID (set to the bridge or Access Point MAC address) or Calling-Station-ID attributes (set to the Supplicant MAC address). As noted in IETF RFC 2865, it is recommended that in this case the User-Name attribute be given the value of Calling-Station-ID.

D.3.6 Framed-Protocol

Since there is no value for IEEE 802 media, the Framed-Protocol attribute is not used by IEEE 802.1X Authenticators.

D.3.7 Framed-IP-Address, Framed-IP-Netmask

IEEE Std 802.1X does not provide a mechanism for IP address assignment. Therefore the Framed-IP-Address and Framed-IP-Netmask attributes can only be used by IEEE 802.1X Authenticators that support IP address assignment mechanisms. Typically this capability is supported by layer 3 devices.

D.3.8 Framed-Routing

The Framed-Routing attribute indicates the routing method for the Supplicant. It is therefore only relevant for IEEE 802.1X Authenticators that act as layer 3 devices and cannot be used by a bridge or Access Point.

D.3.9 Filter-ID

This attribute indicates the name of the filter list to be applied to the Supplicant's session. For use with an IEEE 802.1X Authenticator, it may be used to indicate either layer 2 or layer 3 filters. Layer 3 filters are typically only supported on IEEE 802.1X Authenticators that act as layer 3 devices.

D.3.10 Framed-MTU

This attribute indicates the maximum size of an IP packet that may be transmitted over the wire between the Supplicant and the Authenticator. IEEE 802.1X Authenticators set this to the value corresponding to the relevant IEEE 802 medium and include it in the RADIUS Access-Request. The RADIUS server may send an EAP packet as large as Framed-MTU minus four (4) octets, taking into account the additional overhead for the IEEE 802.1X Version (1), Type(1) and Body Length(2) fields. For EAP over IEEE 802 media, the Framed-MTU values (which do not include LLC/SNAP overhead) and maximum frame length values (not including the preamble) are as shown in Table D-2.

Table D-2—Framed MTU values for IEEE 802 LAN media

IEEE 802 LAN medium	Framed-MTU value	Maximum frame length
Ethernet	1500	1522
IEEE 802.3	1500	1522
IEEE 802.4	8174	8193
IEEE 802.5 (4 Mb/s)	4528	4550
IEEE 802.5 (16 Mb/s)	18173	18200
IEEE 802.5 (100 Mb/s)	18173	18200
IEEE 802.6	9191	9240
IEEE 802.9a	1500	1518
IEEE 802.11	2304[a]	2346
IEEE 802.12 (Ethernet)	1500	1518
IEEE 802.12 (Token Ring)	4502	4528
FDDI	4479	4500

[a]NOTE—The Framed-MTU size for IEEE 802.11 media may change as a result of ongoing work being undertaken in the IEEE 802.11 Working Group. Since some IEEE 802.11 stations cannot handle an MTU larger than 1500 octets, it is recommended that RADIUS servers encountering a NAS-Port-Type value of IEEE 802.11 send EAP packets no larger than 1496 octets.

D.3.11 Framed-Compression

IEEE Std 802.1X does not include compression support so that this attribute is not understood by IEEE 802.1X Authenticators.

D.3.12 Displayable Messages

The Reply-Message attribute, defined in Section 5.18 of IETF RFC 2865, indicates text which may be displayed to the user. This is similar in concept to the EAP Notification Type, defined in IETF RFC 2284. As

noted in IETF RFC 3579, Section 2.6.5, when sending a displayable message to an IEEE 802.1X Authenticator, displayable messages are best sent within EAP-Message/EAP-Request/Notification attribute(s), and not within Reply- Message attribute(s).

D.3.13 Callback-Number, Callback-ID

These attributes are not understood by IEEE 802.1X Authenticators.

D.3.14 Framed-Route

The Framed-Route and Framed-IPv6-Route attributes provide routes that are to be configured for the Supplicant. These attributes are therefore only relevant for IEEE 802.1X Authenticators that act as layer 3 devices, and cannot be understood by a Bridge or Access Point.

D.3.15 State, Class, Proxy-State

These attributes are used for the same purposes as described in IETF RFC 2865.

D.3.16 Vendor-Specific

Vendor-specific attributes are used for the same purposes as described in IETF RFC 2865. The MS-MPPE-Send-Key and MS-MPPE-Recv-Key attributes, described in Section 2.4 of IETF RFC 2548, MAY be used to encrypt and authenticate the RC4 EAPOL-Key descriptor (see 7.6). Examples of the derivation of the MS-MPPE-Send-Key and MS-MPPE-Recv-Key attributes from the master secret negotiated by an EAP method are given in IETF RFC 2716. Details of the EAPOL-Key descriptor are provided in Section 4 of IETF RFC 2716.

D.3.17 Session-Timeout

It is recommended that IEEE 802.1X Authenticators be prepared to receive a Session-Timeout attribute in both an Access-Accept and Access-Challenge.

When sent along in an Access-Accept without a Termination-Action attribute or with a Termination-Action attribute set to Default, the Session-Timeout attribute specifies the maximum number of seconds of service provided prior to session termination.

When sent in an Access-Accept along with a Termination-Action value of RADIUS-Request, the Session-Timeout attribute specifies the maximum number of seconds of service provided prior to reauthentication. In this case, the Session-Timeout attribute is used to load suppTimeout value within the Backend state machine of IEEE 802.1X. When sent with a Termination-Action value of RADIUS-Request, a Session-Timeout value of zero indicates the desire to perform another authentication (possibly of a different type) immediately after the first authentication has successfully completed.

As described in IETF RFC 3579, when sent in an Access-Challenge, this attribute represents the maximum number of seconds that an IEEE 802.1X Authenticator should wait for an EAP-Response before retransmitting. In this case, the Session-Timeout attribute is used to load the suppTimeout constant within the Backend state machine of IEEE 802.1X.

D.3.18 Idle-Timeout

The Idle-Timeout attribute is described in IETF RFC 2865. Since wired IEEE 802 media are always on, the Idle-Timeout attribute is relevant only for IEEE 802.11. It is possible for a wireless device to wander out of range of all Access Points. In this case, the Idle-Timeout attribute indicates the maximum time that an IEEE wireless device may remain idle.

D.3.19 Termination-Action

This attribute indicates what action should be taken when the service is completed. The value RADIUS-Request(1) indicates that reauthentication should occur on expiration of the Session-Time. The value Default (0) indicates that the session should terminate.

D.3.20 Called-Station-Id

For IEEE 802.1X Authenticators, this attribute is used to store the Bridge or Access Point MAC address, represented as an ASCII character string in Canonical format (see IEEE Std 802). For example, “00-10-A4-23-19-C0.” For 802.11 Access Points, the IEEE 802.11 SSID should be appended to the Access Point MAC address, separated from the MAC address with a “:”. For example, “00-10-A4-23-19-C0:AP1”.

D.3.21 Calling-Station-Id

For IEEE 802.1X Authenticators, this attribute is used to store the Supplicant MAC address, represented as an ASCII character string in Canonical format (see IEEE Std 802). For example, “00-10-A4-23-19-C0.”

D.3.22 NAS-Identifier

This attribute contains a string identifying the IEEE 802.1X Authenticator originating the Access-Request.

D.3.23 NAS-Port-Type

For use with IEEE Std 802.1X, NAS-Port-Type values of Ethernet (15), Wireless—IEEE 802.11 (19), Token Ring (20), or FDDI (21) are used.

D.3.24 Port-Limit

This attribute has no meaning when sent to an IEEE 802.1X Authenticator.

D.3.25 Password-Retry

In IEEE Std 802.1X, the Authenticator always transitions to the HELD state after an authentication failure. Thus, this attribute does not make sense for IEEE Std 802.1X.

D.3.26 Connect-Info

This attribute is sent by a bridge or Access Point to indicate the nature of the Supplicant's connection. When sent in the Access-Request, it is recommended that this attribute contain information on the speed of the Supplicant’s connection. For IEEE 802.11, the following format is recommended: “CONNECT 11Mbps 802.11b”. If sent in the Accounting STOP, this attribute may be used to summarize statistics relating to

session quality. For example, in IEEE 802.11, the Connect-Info attribute may contain information on the number of link layer retransmissions. The exact format of this attribute is implementation specific.

D.3.27 EAP-Message

Since IEEE Std 802.1X provides for encapsulation of EAP as described in IETF RFC 2284 and IEEE 802.1X, the EAP-Message attribute defined in IETF RFC 3579 is used to encapsulate EAP packets for transmission from the IEEE 802.1X Authenticator to the Authentication Server. IETF RFC 3579 Section 2.2 describes how the Authentication Server handles invalid EAP packets passed to it by the Authenticator.

D.3.28 Message-Authenticator

As noted in IETF RFC 3579, the Message-Authenticator attribute must be used to protect all packets containing an EAP-Message attribute.

D.3.29 NAS-Port-Id

This attribute is used to identify the IEEE 802.1X Authenticator port that authenticates the Supplicant. The NAS-Port-Id differs from the NAS-Port in that it is a string of variable length, whereas the NAS-Port is a 4-octet value.

D.3.30 Framed-Pool, Framed-IPv6-Pool

IEEE 802.1X does not provide a mechanism for IP address assignment. Therefore the Framed-Pool and Framed-IPv6-Pool attributes can only be used by IEEE 802.1X Authenticators that support IP address assignment mechanisms. Typically, this capability is supported by layer 3 devices.

D.3.31 Tunnel attributes

IETF RFC 2868 defines RADIUS tunnel attributes used for authentication and authorization, and IETF RFC 2867 defines tunnel attributes used for accounting. Where the IEEE 802.1X Authenticator supports tunneling, a compulsory tunnel may be set up for the Supplicant as a result of the authentication.

In particular, it may be desirable to allow a port to be placed into a particular Virtual LAN (VLAN), defined in IEEE Std 8021Q, based on the result of the authentication. This can be used, for example, to allow a wireless host to remain on the same VLAN as it moves within a campus network.

The RADIUS server typically indicates the desired VLAN by including tunnel attributes within the Access-Accept. However, the IEEE 802.1X Authenticator may also provide a hint as to the VLAN to be assigned to the Supplicant by including Tunnel attributes within the Access-Request. For use in VLAN assignment, the following tunnel attributes are used:

a) Tunnel-Type=VLAN (13)

b) Tunnel-Medium-Type=802

c) Tunnel-Private-Group-ID=VLANID

Note that the VLANID is 12-bits, taking a value between 1 and 4094, inclusive. Since the Tunnel-Private-Group-ID is of type String as defined in IETF RFC 2868, for use with IEEE Std 802.1X, the VLANID is encoded as a string, rather than an integer.

When Tunnel attributes are sent, it is necessary to fill in the Tag field. As noted in IETF RFC 2868, Section 3.1:

> The Tag field is one octet in length and is intended to provide a means of grouping attributes in the same packet which refer to the same tunnel. Valid values for this field are 0x01 through 0x1F, inclusive. If the Tag field is unused, it MUST be zero (0x00).

For use with Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Tunnel-Private-Group-ID, Tunnel-Assignment-ID, Tunnel-Client-Auth-ID, or Tunnel-Server-Auth-ID attributes (but not Tunnel-Type, Tunnel-Medium-Type, Tunnel-Password, or Tunnel-Preference), a tag field of greater than 0x1F is interpreted as the first octet of the following field.

Unless alternative tunnel types are provided, (e.g., for IEEE 802.1X Authenticators that may support tunneling but not VLANs), it is only necessary for tunnel attributes to specify a single tunnel. As a result where it is only desired to specify the VLANID, the tag field SHOULD be set to zero (0x00) in all Tunnel attributes. Where alternative tunnel types are to be provided, tag values between 0x01 and 0x1F should be chosen.

D.4 RC4 EAPOL-Key frame

The RC4 EAPOL-Key frame (an Eapol-Key frame with an RC4 Key Descriptor) is created and transmitted by the Authenticator in order to provide media-specific key information. For example, within IEEE Std 802.11 the RC4 EAPOL-Key frame can be used to distribute multicast/broadcast (“default”) keys, or unicast (“key mapping”) keys. The “default” key is the same for all stations within a broadcast domain.The RC4 EAPOL-Key frame is not acknowledged and therefore the Authenticator does not know whether the Supplicant has received it. If it is lost, then the Supplicant and Authenticator will not have the same keying material, and communication will fail. If this occurs, the problem is typically addressed by re-running the authentication.

The RC4 EAPOL-Key frame is sent from the Authenticator to the Supplicant in order to provision the “default” key, and subsequently in order to refresh the “default” key. It may also be used to refresh the key-mapping key. Note that rekey is typically only required with weak ciphersuites such as WEP, defined in IEEE Std 802.11.

Where keys are required, an EAP method that derives keys is typically selected. Therefore, the initial “key mapping” keys can be derived from EAP keying material without requiring the Authenticator to send an RC4 EAPOL-Key frame to the Supplicant. An example of how EAP keying material can be derived and used is presented in IETF RFC 2716.

As described in the paragraphs that follow, the MS-MPPE-Send-Key and MS-MPPE-Recv-Key attributes are defined from the point of view of the Authenticator. From the Supplicant point of reference, the terms are reversed. Thus, the MS-MPPE-Recv-Key on the Supplicant corresponds to the MS-MPPE-Send-Key on the Authenticator, and the MS-MPPE-Send-Key on the Supplicant corresponds to the MS-MPPE-Recv-Key on the Authenticator.

While the RC4 EAPOL-Key frame is defined in Clause 7, a more complete description is provided here as follows:

Protocol Version (see 7.5.3)

The Protocol Version field is one octet. For IEEE Std 802.1X, it contains the value 0x01.

Packet Type (see 7.5.4)

The Packet Type field is one octet, and determines the type of packet being transmitted. For an EAPOL-Key Descriptor, the Packet Type field contains 0x03.

Packet Body Length (see 7.5.5)

The Packet Body Length is two octets, and contains the length of the EAPOL-Key descriptor in octets, not including the Version, Packet Type, and Packet Body Length fields.

Descriptor Type (see 7.6.1)

The Descriptor Type field is a single octet. The Key descriptor is defined differently for each Descriptor Type; for the RC4 Key Descriptor, this field takes the value 0x01.

Key Length (see 7.6.3.1)

The Key Length field is two octets. If Packet Body Length = 44 + Key Length, then the Key Field contains the key in encrypted form, of length Key Length. This is 5 octets (40 bits) for WEP, and 13 octets (104 bits) for WEP-128. If Packet Body Length = 44, then the Key field is absent, and Key Length represents the number of least significant octets from the MS-MPPE-Send-Key attribute to be used as the keying material.

Replay Counter (see 7.6.3.2)

The Replay Counter field is 8 octets. It does not repeat within the life of the keying material used to encrypt the Key field and compute the Key Message Digest field. A 64-bit NTP timestamp may be used as the Replay Counter.

Key IV (see 7.6.3.3)

The Key IV field is 16 octets and includes a 128-bit nonce (see 3.3.1).

Key Index (see 7.6.3.4)

The high order bit (bit 8) of this field contains the Key flag (F), which is a single bit, describing the type of key that is included in the Key field. Values are:

0 = for broadcast (default key)

1 = for unicast (key mapping key)

The Key Index value is encoded in the remaining 7 bits.

Key Message Digest (see 7.6.3.5)

The Key Message Digest field is 16 octets. It contains an HMAC-MD5 message integrity check computed over the EAPOL-Key descriptor, starting from the Version field, with the Key field filled in if present, but

with the Key Message Digest field set to zero. For the computation, the 32 octet (256 bit) MS-MPPE-Send-Key is used as the HMAC-MD5 key.

Key (see 7.6.3.6)

If Packet Body Length = 44 + Key Length, then the Key Field contains the key in encrypted form, of length Key Length. If Packet Body Length = 44, then the Key field is absent, and the least significant Key Length octets from the MS-MPPE-Send-Key attribute (see IETF RFC 2548) is used as the keying material. Where the Key field is encrypted using RC4, the RC4 encryption key used to encrypt this field is formed by concatenating the 16 octet (128 bit) Key-IV field with the 32 octet MS-MPPE-Recv- Key attribute. This yields a 48 octet RC4 key (384 bits).

D.5 Security considerations

Since this standard describes the use of RADIUS for purposes of authentication authorization and accounting in IEEE 802.1X-enabled networks, it is vulnerable to all of the threats that are present in other RADIUS applications, with one exception. For a discussion of these threats, see IETF RFC 2607 and IETF RFC 3579.

Vulnerabilities include:

a) Packet modification or forgery

b) Dictionary attacks

c) Known plaintext attacks

d) Replay

e) Outcome mismatches

f) IEEE 802.11 integration

g) Key management issues

D.5.1 Packet modification or forgery

RADIUS, defined in IETF RFC 2865, does not require all Access-Requests to be authenticated or integrity protected. However, IEEE Std 802.1X is based on EAP. As described in IETF RFC 3579:

> The Message-Authenticator attribute MUST be used to protect all Access-Request, Access-Challenge, Access-Accept, and Access-Reject packets containing an EAP-Message attribute.

As a result, when used with IEEE Std 802.1X, all RADIUS packets must be authenticated and integrity protected. In addition, as described in IETF RFC 3579, Section 4.2:

> To address the security vulnerabilities of RADIUS/EAP, implementations of this specification should support IPsec (see IETF RFC 2401) along with IKE (see IETF RFC 2409) for key management. IPsec ESP (see IETF RFC 2406) with non-null transform SHOULD be supported, and IPsec ESP with a non-null encryption transform and authentication support SHOULD be used to provide per-packet confidentiality, authentication, integrity and replay protection. IKE SHOULD be used for key management.

D.5.2 Dictionary attacks

As discussed in IETF RFC 3579 Section 4.3.3, the RADIUS shared secret is vulnerable to offline dictionary attack, based on capture of the Response Authenticator or Message-Authenticator attribute. In order to decrease the level of vulnerability, IETF RFC 2865 Section 3 recommends:

> The secret (password shared between the client and the RADIUS server) SHOULD be at least as large and unguessable as a well-chosen password. It is preferred that the secret be at least 16 octets.

In addition, the risk of an offline dictionary attack can be further mitigated by employing IPsec ESP with non-null transform in order to encrypt the RADIUS conversation, as described in IETF RFC 3579, Section 4.2.

D.5.3 Known plaintext attacks

Since IEEE Std 802.1X is based on EAP, which does not support PAP, the RADIUS User-Password attribute is not used to carry hidden user passwords. The hiding mechanism utilizes MD5, defined in IETF RFC 1321, in order to generate a key stream based on the RADIUS shared secret and the Request Authenticator. Where PAP is in use, it is possible to collect key streams corresponding to a given Request Authenticator value, by capturing RADIUS conversations corresponding to a PAP authentication attempt using a known password. Since the User-Password is known, the key stream corresponding to a given Request Authenticator can be determined and stored.

The vulnerabilities are described in detail within IETF RFC 3579, Section 4.3.5. Even though IEEE 802.1X Authenticators do not support PAP authentication, a security vulnerability can still exist where the same RADIUS shared secret is used for hiding User-Password as well as other attributes. This can occur, for example, if the same RADIUS proxy handles authentication requests for both IEEE 802.1X (which may hide the Tunnel-Password, MS-MPPE-Send-Key and MS-MPPE-Recv-Key attributes) and GPRS (which may hide the User-Password attribute).

The threat can be mitigated by protecting RADIUS with IPsec ESP with non-null transform, as described in IETF RFC 3579, Section 4.2. In addition, the same RADIUS shared secret must not be used for both IEEE 802.1X authentication and PAP authentication.

D.5.4 Replay

As noted in IETF RFC 3579 Section 4.3.5, the RADIUS protocol provides only limited support for replay protection. Replay protection for RADIUS authentication and accounting can be provided by enabling IPsec replay protection with RADIUS, as described in IETF RFC 3579, Section 4.2.

As with the Request Authenticator, for use with IEEE 802.1X Authenticators, the Acct-Session-Id should be globally and temporally unique.

D.5.5 Outcome mismatches

IETF RFC 3579, Section 2.6.3, discusses the issues that arise when the EAP packet encapsulated in an EAP-Message attribute does not agree with the RADIUS Packet Type. For example, an EAP Success packet might be encapsulated within an Access-Reject, an EAP Failure within an Access-Accept, or an EAP Success or Failure might be sent with an Access-Challenge.

As described in IETF RFC 3579, Section 2.6.3, these conflicting messages are likely to cause confusion. To ensure that access decisions made by IEEE 802.1X Authenticators conform to the wishes of the RADIUS

server, it is necessary for the Authenticator to make the decision solely based on the authentication result (Access-Accept/Reject) and not based on the contents of EAP-Message attributes, if present.

D.5.6 IEEE 802.11 integration

IEEE Std 802.1X was developed for use on wired IEEE 802 networks such as Ethernet, and therefore does not describe how to securely adapt IEEE Std 802.1X for use with IEEE Std 802.11. This is left to the enhanced security specification under development within IEEE Std 802.11.[16]

For example, IEEE Std 802.1X does not specify whether authentication occurs prior to, or after, association, nor how the derived keys are used within various cyphersuites. It also does not specify ciphersuites addressing the vulnerabilities discovered in WEP, described in [B2], [B1], [B3], and [B8]. IEEE Std 802.1X only defines an authentication framework, leaving the definition of the authentication methods to other documents, such as IETF RFC 2716.

Since IEEE Std 802.1X does not address IEEE 802.11 integration issues, implementors are strongly advised to consult the additional IEEE 802.11 security specification for guidance on how to adapt IEEE 802.1X for use with IEEE Std 802.11. For example, it is likely that the IEEE 802.11 enhanced security specification will define its own IEEE 802.11 key hierarchy as well as new EAPOL-Key descriptors.

D.5.7 Key management issues

The EAPOL-Key descriptor described in D.4 is likely to be deprecated in the future when the IEEE 802.11 enhanced security group completes its work. Known security issues include:

a) **Default key-only support.** IEEE 802.1X enables the derivation of per-station unicast keys, known in IEEE Std 802.11 as “key mapping keys.” Keys used to encrypt multicast/broadcast traffic are known as “default keys”. However, in some IEEE 802.11 implementations, the unicast keys derived as part of the EAP authentication process are used solely in order to encrypt, authenticate, and integrity protect the EAPOL-Key descriptor, as described in D.4. These implementations only support use of default keys (ordinarily only used with multicast/broadcast traffic) to secure all traffic, unicast or multicast/broadcast, resulting in inherent security weaknesses.

   Where per-station key-mapping keys (e.g., unicast keys) are unsupported, any station possessing the default key can decrypt traffic from other stations or impersonate them. When used along with a weak cipher (e.g., WEP), implementations supporting only default keys provide more material for attacks such as those described in [B3] and [B8]. If in addition the default key is not refreshed periodically, IEEE 802.1X dynamic key derivation provides little or no security benefit. For an understanding of the issues with WEP, see [B2], [B1], [B3], and [B8].

b) **Reuse of keying material.** The EAPOL-Key descriptor specified in Clause 7 uses the same keying material (MS-MPPE-Recv-Key) both to encrypt the Key field within the EAPOL-Key descriptor, as well as to encrypt data passed between the station and access point. Multi-purpose keying material is frowned upon, since multiple uses can leak information helpful to an attacker

c) **Weak algorithms.** The algorithm used to encrypt the Key field within the EAPOL-Key descriptor is similar to the algorithm used in WEP, and as a result, shares some of the same weaknesses. As with WEP, the RC4 stream cipher is used to encrypt the key. As input to the RC4 engine, the IV and key are concatenated rather than being combined within a mixing function. As with WEP, the IV is not a counter, and therefore there is little protection against reuse.

[16] This enhanced security specification is being developed under project P802.11i.

As a result of these vulnerabilities, implementors intending to use the EAPOL-Key descriptor described in this document are urged to consult the IEEE 802.11 enhanced security specification for a more secure alternative. It is also advisable to consult the evolving literature on WEP vulnerabilities, in order to better understand the risks, as well as to obtain guidance on setting an appropriate re-keying interval.

D.6 IANA considerations

This specification requires the assignment of new values to existing RADIUS attributes.[17] These include the values shown in Table D-1.

Table D-1—New RADIUS attribute values

Attribute	Values required
NAS-Port-Type	Token-Ring (20), FDDI (21)
Tunnel-Type	VLAN (13)
Acct-Terminate-Cause	Supplicant Restart (19) Reauthentication Failure (20) Port Reinitialized (21) Port Administratively Disabled (22)

D.7 Tables of Attributes

Table D-2 provides a guide to which attributes may be sent and received as part of IEEE 802.1X Authentication. For each attribute, the reference provides the definitive information on usage.

X in the IEEE 802.1X column indicates that the attribute may be used with IEEE 802.1X Authentication.

L3 in the IEEE 802.1X column indicates that the attribute is typically implemented only by Authenticators with layer 3 capabilities.

[17]The IANA RADIUS registry for RADIUS Type assignments can be found at http://www.iana.org/assignments/radius-types

Table D-2—RADIUS attributes in IEEE 802.1X

IEEE 802.1X	#	Attribute
X	1	User-Name (see IETF RFC 2865)
	2	User-Password (see IETF RFC 2865)
	3	CHAP-Password (see IETF RFC 2865)
X	4	NAS-IP-Address (see IETF RFC 2865)
X	5	NAS-Port (see IETF RFC 2865)
X	6	Service-Type (see IETF RFC 2865)
	7	Framed-Protocol (see IETF RFC 2865)
L3	8	Framed-IP-Address (see IETF RFC 2865)
L3	9	Framed-IP-Netmask (see IETF RFC 2865)
L3	10	Framed-Routing (see IETF RFC 2865)
X	11	Filter-Id (see IETF RFC 2865)
X	12	Framed-MTU (see IETF RFC 2865)
	13	Framed-Compression (see IETF RFC 2865)
L3	14	Login-IP-Host (see IETF RFC 2865)
L3	15	Login-Service (see IETF RFC 2865)
L3	16	Login-TCP-Port (see IETF RFC 2865)
	18	Reply-Message (see IETF RFC 2865)
	19	Callback-Number (see IETF RFC 2865)
	20	Callback-Id (see IETF RFC 2865)
L3	22	Framed-Route (see IETF RFC 2865)
L3	23	Framed-IPX-Network (see IETF RFC 2865)
X	24	State (see IETF RFC 2865)
X	25	Class (see IETF RFC 2865)
X	26	Vendor-Specific (see IETF RFC 2865)
X	27	Session-Timeout (see IETF RFC 2865)
X	28	Idle-Timeout (see IETF RFC 2865)
X	29	Termination-Action (see IETF RFC 2865)

Table D-2—RADIUS attributes in IEEE 802.1X *(continued)*

IEEE 802.1X	#	Attribute
X	30	Called-Station-Id (see IETF RFC 2865)
X	31	Calling-Station-Id (see IETF RFC 2865)
X	32	NAS-Identifier (see IETF RFC 2865)
X	33	Proxy-State (see IETF RFC 2865)
	34	Login-LAT-Service (see IETF RFC 2865)
	35	Login-LAT-Node (see IETF RFC 2865)
	36	Login-LAT-Group (see IETF RFC 2865)
L3	37	Framed-AppleTalk-Link (see IETF RFC 2865)
L3	38	Framed-AppleTalk-Network (see IETF RFC 2865)
L3	39	Framed-AppleTalk-Zone (see IETF RFC 2865)
X	40	Acct-Status-Type (see IETF RFC 2866)
X	41	Acct-Delay-Time (see IETF RFC 2866)
X	42	Acct-Input-Octets (see IETF RFC 2866)
X	43	Acct-Output-Octets (see IETF RFC 2866)
X	44	Acct-Session-Id (see IETF RFC 2866)
X	45	Acct-Authentic (see IETF RFC 2866)
X	46	Acct-Session-Time (see IETF RFC 2866)
X	47	Acct-Input-Packets (see IETF RFC 2866)
X	48	Acct-Output-Packets (see IETF RFC 2866)
X	49	Acct-Terminate-Cause (see IETF RFC 2866)
X	50	Acct-Multi-Session-Id (see IETF RFC 2866)
X	51	Acct-Link-Count (see IETF RFC 2866)
X	52	Acct-Input-Gigawords (see IETF RFC 2869)
X	53	Acct-Output-Gigawords (see IETF RFC 2869)
X	55	Event-Timestamp (see IETF RFC 2869)
	60	CHAP-Challenge (see IETF RFC 2865)
X	61	NAS-Port-Type (see IETF RFC 2865)

Table D-2—RADIUS attributes in IEEE 802.1X *(continued)*

IEEE 802.1X	#	Attribute
	62	Port-Limit (see IETF RFC 2865)
	63	Login-LAT-Port (see IETF RFC 2865)
X	64	Tunnel-Type (see IETF RFC 2868)
X	65	Tunnel-Medium-Type (see IETF RFC 2868)
L3	66	Tunnel-Client-Endpoint (see IETF RFC 2868)
L3	67	Tunnel-Server-Endpoint (see IETF RFC 2868)
L3	68	Acct-Tunnel-Connection (see IETF RFC 2867)
L3	69	Tunnel-Password (see IETF RFC 2868)
	70	ARAP-Password (see IETF RFC 2869)
	71	ARAP-Features (see IETF RFC 2869)
	72	ARAP-Zone-Access (see IETF RFC 2869)
	73	ARAP-Security (see IETF RFC 2869)
	74	ARAP-Security-Data (see IETF RFC 2869)
	75	Password-Retry (see IETF RFC 2869)
	76	Prompt (see IETF RFC 2869)
X	77	Connect-Info (see IETF RFC 2869)
X	78	Configuration-Token (see IETF RFC 2869)
X	79	EAP-Message (see IETF RFC 3579)
X	80	Message-Authenticator (see IETF RFC 3579)
X	81	Tunnel-Private-Group-ID (see IETF RFC 2868)
L3	82	Tunnel-Assignment-ID (see IETF RFC 2868)
X	83	Tunnel-Preference (see IETF RFC 2868)
	84	ARAP-Challenge-Response (see IETF RFC 2869)
X	85	Acct-Interim-Interval (see IETF RFC 2869)
X	86	Acct-Tunnel-Packets-Lost (see IETF RFC 2867)
X	87	NAS-Port-Id (see IETF RFC 2869)

Table D-2—RADIUS attributes in IEEE 802.1X *(continued)*

IEEE 802.1X	#	Attribute
L3	88	Framed-Pool (see IETF RFC 2869)
L3	90	Tunnel-Client-Auth-ID (see IETF RFC 2868)
L3	91	Tunnel-Server-Auth-ID (see IETF RFC 2868)
X	95	NAS-IPv6-Address (see IETF RFC 3162)
	96	Framed-Interface-Id (see IETF RFC 3162)
L3	97	Framed-IPv6-Prefix (see IETF RFC 3162)
L3	98	Login-IPv6-Host (see IETF RFC 3162)
L3	99	Framed-IPv6-Route (see IETF RFC 3162)
L3	100	Framed-IPv6-Pool (see IETF RFC 3162)
X	101	Error-Cause (see IETF RFC 3576)

Annex E

(informative)

PAE state machine interface with higher layers: EAP and AAA

E.1 Introduction

This annex defines the interface to the higher layer assumed by the PAE state machines. An example of such a higher layer is currently being defined by the EAP working group within the IETF. Figure 8-1 and Figure 8-8 show the signals between the PAE state machines and the higher layers in a diagrammatic format. This material is included here for information only; it does not constitute a requirement of this standard.

E.2 Supplicant

The PAE Supplicant state machines communicate with a single higher layer entity: EAP. On the Supplicant, PAE's role is to transport EAP frames between the network and the higher layer and to optionally control port access based on the result of the authentication exchange. The Supplicant state machines do this without examining the EAP header in the frame. The interface between the PAE Supplicant state machines and EAP is defined by a set of state machine variables: portEnabled, eapRestart, eapReq, eapResp, eapNoResp, eapSuccess, eapFail.

E.2.1 portEnabled

The Supplicant state machine expects the higher layer to watch this signal and begin an initialization process when the signal is asserted. This ensures that both the PAE state machine and the higher layer are in sync at initialization time. It is expected that the higher layer will reset both the eapSuccess and eapFail at this time.

E.2.2 eapRestart

The Supplicant state machine sets eapRestart to indicate to the higher layer that the PAE is requesting it to restart. It is expected that the higher layer will reinitialize itself to the same state achieved when the port first becomes enabled. The higher layer must reset this signal to indicate it has completed its own restart process. This signal is required to inform the higher layer of a PAE event that should cause the termination of in-progress or outstanding authentications. It is expected that the higher layer will reset both the eapSuccess and eapFail signals at this time.

E.2.3 eapReq

The Supplicant state machine sets this signal to indicate to the higher layer that there is a new EAP frame to be processed. It is expected that the higher layer will collect the EAP frame, process it within its current state and then set either eapResp or eapNoResp depending upon the next action the PAE Supplicant should take. The higher layer resets the eapReq variable to indicate it has completed processing of the EAP frame.

E.2.4 eapResp

This signal is set by the higher layer after it has received an EAP Request from the PAE layer for processing and it has an EAP Response frame available for the PAE layer to transmit. After the higher layer sets this

signal, it is assumed that it will also reset the eapReq signal to indicate it has completed processing of the EAP Request that led to this response. The eapResp signal will be reset by the PAE layer after it has transmitted the EAP Response.

E.2.5 eapNoResp

This signal is set by the higher layer after it has received an EAP Request from the PAE layer for processing and has decided that it will not need to transmit an EAP Response frame. This typically occurs if there is an error in the EAP frame or some anomaly has occurred in the EAP authentication exchange. After the higher layer sets this signal, it should reset the eapReq signal to indicate it has completed processing of the EAP Request that led to this response. This signal will be reset by the PAE layer to acknowledge that it will send no response.

E.2.6 eapSuccess

The eapSuccess signal is set by the higher layer to indicate that the EAP authentication exchange has completed with a successful outcome. This will cause the PAE state machine to initiate other processing that may result in entering the authenticated state. The higher layer should set this signal in conjunction with the eapNoResp signal. This signal is reset by the higher layer during its initialization of state.

E.2.7 eapFail

This signal is set by the higher layer to indicate that the EAP authentication exchange has completed with an unsuccessful outcome. This will cause the PAE state machine to initiate other processing that will result in entering the held state. The higher layer should set this signal in conjunction with the eapNoResp signal. This signal is reset by the higher layer during its initialization of state.

E.3 Authenticator

The PAE Authenticator state machines communicate with a higher layer entity, that manages EAP and AAA functionality. On the Authenticator, the PAE's role is to transport EAP frames between the Supplicant and the Authenticator's higher-layer entity and to control port access based on the result of the authentication exchange. The Authenticator state machines do this without examining the EAP header in the frame. The interface between the PAE Authenticator state machines and the higher layer is defined by a set of state machine variables: portEnabled, eapRestart, eapReq, eapNoReq, eapResp, eapSuccess, eapFail, and keyAvailable.

E.3.1 portEnabled

The higher layer is expected to initialize itself when this signal becomes true. This ensures that both the PAE state machine and the higher layer is in sync at initialization time. The higher layer is expected to reset eapSuccess and eapFail when portEnabled is initially set true.

E.3.2 eapRestart

The Authenticator state machine sets this signal to indicate to the higher layer that PAE is restarting. It is expected that the higher layer will reset its state to remain synchronized with the PAE state machine. The higher layer must reset this signal to indicate it has completed its own restart. This signal is required to inform the higher layer of a PAE event that should cause the termination of in-progress or outstanding

authentications. The higher layer must reset eapSuccess and eapFail after eapRestart is set true by the PAE layer.

E.3.3 eapReq

The higher layer sets this signal to indicate that a new EAP frame that requires a response is ready to be transmitted to the Supplicant. This signal is not set for EAP frames that do not require a response (e.g., EAP-SUCCESS and EAP-FAILURE). Following a restart, the higher layer must create an initial EAP Request to be sent and set this signal in order for the PAE state machine to move forward. The PAE state machine will reset this signal after it has transmitted the EAP frame.

E.3.4 eapResp

The PAE Authenticator state machine sets this signal when a new EAP frame has been received from the Supplicant and is ready for processing by the higher layer. The higher layer is expected to process the response and assert one of the following signals: eapSuccess, eapFail, eapReq, or eapNoReq. The higher layer should then reset the eapResp signal to indicate it has completed the processing of the EAP frame.

E.3.5 eapNoReq

The higher layer sets this signal to inform the PAE Authenticator state machine to wait for another EAP Response from the Supplicant. This typically occurs if there was an error in the previous EAP Response frame. After setting this value true, the higher layer should reset the eapResp signal. The PAE Authenticator state machine resets the eapNoReq signal to acknowledge it was received and that the PAE state machine is now awaiting a new EAP Response.

E.3.6 eapSuccess

The higher layer should set this signal when a AAA ACCESS-ACCEPT message has been received from the Authentication Server that indicates the authentication process has completed successfully.

NOTE—This signal does not indicate whether the ACCESS-ACCEPT contains an EAP-SUCCESS or not. The PAE Authenticator state machine will transmit the EAP frame contained within the ACCESS-ACCEPT and then begin taking steps that may lead to entering the authenticated state. The higher layer is expected to reset eapSuccess any time it is initialized.

E.3.7 eapFail

The higher layer should set this signal to indicate that a AAA ACCESS-REJECT message has been received that indicates the authentication process has completed unsuccessfully.

NOTE—This signal does not indicate whether the ACCESS-REJECT message contains an EAP-FAILURE or not. The PAE Authenticator state machine will transmit the EAP frame contained within the ACCESS-REJECT and then initiate the steps necessary to enter the held state. The higher layer should reset this value any time it is initialized.

E.3.8 eapTimeout

The higher layer should set this signal to indicate that it has waited too long to receive a new EAP Request from the Authentication Server. The PAE Authenticator state machine will begin the process of aborting the current authentication exchange and will restart a new authentication. The higher layer should reset this signal whenever it is initialized.

E.3.9 keyAvailable

The higher layer should set this signal when the Authentication Server has made available any keying material necessary to generate an EAPOL-Key message. Once this signal is set the PAE key transmit machine will gather the key material and begin the process of making the link secure. The PAE key transmit machine will reset this signal once the link has been made secure.

Annex F

(informative)

Bibliography

[B1] Arbaugh, W., Shankar, N., Wan, J.Y.C., "Your 802.11 Wireless Network has No Clothes", Department of Computer Science, University of Maryland, College Park, March 2001.

[B2] Borisov, N., Goldberg, I., Wagner, D., "Intercepting Mobile Communications: The Insecurity of 802.11", ACM.SIGMOBILE, Seventh Annual International Conference on Mobile Computing and Networking, July 2001, Rome, Italy.

[B3] Fluhrer, S., Mantin, I., Shamir, A., "Weaknesses in the Key Scheduling Algorithm of RC4", Eighth Annual Workshop on Selected Areas in Cryptography, Toronto, Canada, August 2001.

[B4] IETF RFC 3588, Diameter Base Protocol, P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, September 2003.

[B5] IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition.

[B6] IEEE Std 802.1X-2001, IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control.

[B7] ISO/IEC 14882:2003, Programming languages—C++.

[B8] Stubblefield, A., Ioannidis, J., Rubin, A., "Using the Fluhrer, Mantin and Shamir Attack to Break WEP", 2002 NDSS Conference.