IEEE STANDARDS ASSOCIATION

IEEE Standard for Local and Metropolitan Area Networks—

Bridges and Bridged Networks

Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

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

IEEE Std 802.1Qcc™-2018
(Amendment to
IEEE Std 802.1Q™-2018
as amended by
IEEE Std 802.1Qcp™-2018)

IEEE Std 802.1Qcc™-2018
(Amendment to
IEEE Std 802.1Q™-2018
as amended by
IEEE Std 802.1Qcp™-2018)

IEEE Standard for
Local and Metropolitan Area Networks—

Bridges and Bridged Networks

Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 14 June 2018

IEEE-SA Standards Board

Abstract: Enhancements to the configuration of time-sensitive streams are provided by this amendment to IEEE Std 802.1Q-2018.

Keywords: amendment, Bridged Local Area Networks, IEEE 802©, IEEE 802.1Q™, IEEE 802.1Qcc™, SRP, Stream Reservation Protocol, MSRP, Multiple Stream Registration Protocol, Time-Sensitive Networking, TSN

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

PDF: ISBN 978-1-5044-5064-5 STD23223
Print: ISBN 978-1-5044-5065-2 STDPD23223

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at https://standards.ieee.org/IPR/disclaimers.html.

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

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

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

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

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

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

Translations

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

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

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

Comments on standards should be submitted to the following address:

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

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

Photocopies

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. A current IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

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

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at https://ieeexplore.ieee.org or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE-SA Website at https://standards.ieee.org.

Errata

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

Patents

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

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

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

Glenn Parsons, *Chair*
John Messenger, *Vice Chair* and *Acting Chair*
Jessy V. Rouyer, *Acting Vice Chair*
János Farkas, *Chair, Time-Sensitive Networking Task Group*
Rodney Cummings, *Editor*

Ralf Assmann
Shenghua Bao
Jens Bierschenk
Steinar Bjornstad
Christian Boiger
Paul Bottorff
Radhakrishna Canchi
David Chen
Feng Chen
Weiying Cheng
Paul Congdon
Hesham Elbakoury
Norman Finn
Geoffrey Garner
Eric W. Gray
Craig Gunther
Marina Gutierrez
Stephen Haddock
Mark Hantel
Lokesh Kabra
Michael Karl
Stephan Kehrer
Hajime Koto
Christophe Mangin
Scott Mansfield
James McIntosh
Tero Mustala
Tomoki Ohsawa
Donald R. Pannell
Walter Pienciak
Michael Potts
Wei Qiu
Karen Randall
Maximilian Riegel
Atsushi Sato
Frank Schewe
Michael Seaman
Johannes Specht
Patricia Thaler
Paul Unbehagen
Xinyuan Wang
Tongtong Wang
Hao Wang
Karl Weber
Brian Weis
Jordon Woods
Takahiro Yamaura
Xiang Yu
Nader Zein

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

Thomas Alexander
Richard Alfvin
Butch Anton
Stefan Aust
Gordon Bechtel
Christian Boiger
Nancy Bravin
Dietmar Bruckner
Demetrio Bucaneg
Ashley Butterworth
William Byrd
Steven Carlson
Juan Carreon
Keith Chow
Charles Cook
Rodney Cummings
Sourav Dutta
János Farkas
Norman Finn
Avraham Freedman
Eric W. Gray
Randall Groves
Craig Gunther
Stephen Haddock
Mark Hantel
Marco Hernandez
Werner Hoelzl
Rita Horner
Noriyuki Ikeuchi
Atsushi Ito
Raj Jain
SangKwon Jeong
Piotr Karocki
Stephan Kehrer
Stuart Kerry
Yongbum Kim
Hyeong Ho Lee
James Lepp
Jon Lewis
Michael Lynch
Elvis Maculuba
Arthur Marris
Richard Mellitz
Charles Moorwood
Charles Ngethe
Nick S. A. Nikjoo
Paul Nikolich
Satoshi Obara
Bansi Patel
Adee Ran
Alon Regev
Maximilian Riegel
Robert Robinson
Benjamin Rolfe
Jessy V. Rouyer
Michael Seaman
Daniel Smith
Thomas Starai
Walter Struppler
Mark-Rene Uchida
Dmitri Varsanofiev
George Vlantis
Lisa Ward
Andreas Wolf
Oren Yuen
Zhen Zhou

When the IEEE-SA Standards Board approved this amendment on 14 June 2018, it had the following membership:

Jean-Philippe Faure, *Chair*
Gary Hoffman, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
Guido R. Hiertz
Christel Hunter
Joseph L. Koepfinger*
Thomas Koshy
Hung Ling
Dong Liu
Xiaohui Liu
Kevin Lu
Daleep Mohla
Andrew Myles
Paul Nikolich
Ronald C. Petersen
Annette D. Reilly
Robby Robson
Dorothy Stanley
Mehmet Ulema
Phil Wennblom
Philip Winston
Howard Wolfman
Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1Qcc-2018, IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements.

This amendment to IEEE Std 802.1Q-2018 provides enhancements to the configuration of time-sensitive streams.

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

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854-4141
USA

Contents

Figures

Tables

IEEE Standard for
Local and Metropolitan Area Networks—

Bridges and Bridged Networks

Amendment 31:
Stream Reservation Protocol (SRP)
Enhancements and Performance Improvements

(This amendment is based on IEEE Std 802.1Q™-2018, as amended by IEEE Std 802.1Qcp™-2018.)

NOTE—Text shown in ***bold italics*** in this amendment defines the editing instructions necessary to change to the base standard. Three editing instructions are used: change, delete, and insert. ***Change*** is used to make changes to existing material. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ to remove old material and underscore to add new material. ***Delete*** removes existing material. ***Insert*** adds new material without changing the existing material. Insertions, however, may require renumbering. If so, renumbering instructions are given in the editing instruction. Editing instructions, change markings, and this NOTE will not be carried over into future editions of IEEE Std 802.1Q because the changes will be incorporated into the base standard.[1]

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

1. Overview

1.3 Introduction

Insert the following text at the end of 1.3:

This standard specifies enhancements to protocols, procedures, and managed objects for the configuration of network resources for time-sensitive (i.e., bounded latency) applications. The enhancements address Time-Sensitive Networking (TSN) application requirements beyond audio/video (AV) traffic. To this end, it

cl) Specifies a software interface between the user (i.e., time-sensitive application) and network components, such that the user provides Stream requirements (e.g., for bounded latency), and the network configures resources from Talker to Listeners to meet those requirements. This user/network interface (UNI) is specified as an information model that can be applied to any protocol.
cm) Specifies three models for the UNI: fully distributed, centralized network/distributed user, and fully centralized.
cn) Specifies enhancements to the Stream Reservation Protocol (SRP), using a new application version, MSRPv1. MSRPv1 integrates the UNI TLVs for the benefits of enhanced configuration. For compatibility, MSRPv1 translates to the previous version (MSRPv0).
co) Specifies enhancements to the managed objects for forwarding and queuing enhancements for time-sensitive streams (FQTSS).
cp) Specifies enhancements to the managed objects for SRP.
cq) Specifies managed objects for configuration of Bridges by a Centralized Network Configuration (CNC) component.

2. Normative references

Insert the following reference into Clause 2 in alphanumeric order:

IETF RFC 7950, The YANG 1.1 Data Modeling Language, August 2016.[2]

[2] IETF documents (i.e., RFCs) are available from the Internet Engineering Task Force (https://www.rfc-editor.org/).

3. Definitions

Change the following definition as shown:

3.259 stream reservation (SR) class: A traffic class whose bandwidth can be reserved for ~~audio/video (AV) traffic~~ time-sensitive streams using the Stream Reservation Protocol (SRP). A priority value is associated with each SR class. SR classes are denoted by consecutive letters of the alphabet, starting with A and continuing for up to seven classes.

Insert the following terms into Clause 3 in alphabetic order, number them appropriately, and renumber the subsequent terms in the clause accordingly:

3.x Centralized Network Configuration (CNC): A centralized component that configures network resources on behalf of TSN applications (users).

3.x Centralized User Configuration (CUC): A centralized entity that discovers end stations, retrieves end station capabilities and user requirements, and configures TSN features in end stations. The protocols that the CUC uses for communication with end stations are specific to the user application, not specified in this standard. A CUC exchanges information with a CNC in order to configure TSN features on behalf of its end stations.

3.x time-aware: An adjective to describe use of time that is synchronized with other stations using a protocol (e.g., IEEE Std 802.1AS).

4. Abbreviations

Insert the following abbreviations into Clause 4 in alphabetic order:

CNC	Centralized Network Configuration
CUC	Centralized User Configuration
TSN	Time-Sensitive Networking

5. Conformance

5.2 Conformant components and equipment

Insert the following item at the end of the lettered list in 5.2:

r) TSN CNC station (5.29)

5.4 VLAN Bridge component requirements

5.4.1 VLAN Bridge component options

Insert the following item at the end of the lettered list in 5.4.1:

ae) Support Stream reservation remote management (5.4.1.10).

5.4.1.5 Forwarding and queuing enhancements for time-sensitive streams (FQTSS)—requirements

Insert the following items at the end of the lettered list in 5.4.1.5:

h) Support the classMeasurementInterval attribute (12.20.1).
i) Support the SR Class to Priority Mapping Table (12.20.4).

Insert the following subclause (5.4.1.10) after 5.4.1.9:

5.4.1.10 Stream reservation remote management (optional)

A VLAN-aware Bridge component implementation that conforms to the provisions of this standard for Stream reservation remote management shall

a) Support the management functionality defined in Clause 12.
b) Support the use of a remote management protocol. Bridges claiming to support remote management shall state the following:
    1) Which remote management protocol standard(s) or specification(s) are supported for the server implementation.
    2) Which standard(s) or specification(s) for managed object definitions and encodings are supported for use by the remote management protocol.
c) Support TE-MSTID (5.5.2).
d) Support the entities for Stream reservation remote management (12.32).
e) Support the adminIdleSlope attribute (12.20.1).

5.5 C-VLAN component conformance

5.5.1 C-VLAN component options

Insert the following item at the end of the lettered list in 5.5.1:

d) Support TE-MSTID (5.5.2).

Insert the following subclause (5.5.2) after 5.5.1:

5.5.2 TE-MSTID (optional)

A C-VLAN component implementation that conforms to the provisions of this standard for TE-MSTID shall

a) Disable learning for a set of Customer VLAN Identifiers (C-VIDs) allocated to the Traffic Engineering Multiple Spanning Tree Instance Identifier (TE-MSTID) as specified in 8.4 and 8.9, with the exception that support for PBB-TE is not required; and
b) Discard frames with unregistered destination addresses for C-VIDs allocated to the TE-MSTID (8.8.1).

A C-VLAN component implementation that conforms to the provisions of this standard for TE-MSTID may

c) Allow control of all C-VIDs by an external agent, conforming to the Reserved VID values of Table 9-2.

Insert the following subclause (5.29) after 5.28:

5.29 TSN CNC station requirements

This subclause defines the conformance requirements for a station that supports the Time-Sensitive Networking (TSN) Centralized Network Configuration (CNC) requirements (46.1.3). The TSN CNC station component is implemented within an end station or Bridge.

A TSN CNC station implementation that conforms to the provisions of this standard shall:

a) Support the use of a remote management protocol. The TSN CNC claiming to support remote management shall state the following:
    1) Which remote management protocol standard(s) or specification(s) are supported for the client implementation.
    2) Which standard(s) or specification(s) for managed object definitions and encodings are supported for use by the remote management protocol.
b) Support the managed object definitions and encodings for Stream reservation remote management (12.32).
c) Support the use of at least one protocol for User/network configuration information that complies with the requirements for protocol integration defined in 46.2. The TSN CNC shall state which User/network configuration information protocol standard(s) or specification(s) are supported.
d) If a YANG-based protocol is supported by the TSN CNC for the User/network configuration information, that protocol shall use the YANG module specified in 46.3.
e) If SRP (Clause 35) is supported by the TSN CNC for the User/network configuration information, the TSN CNC shall support MRP External Control (12.32.4).

10. Multiple Registration Protocol (MRP) and Multiple MAC Registration Protocol (MMRP)

10.6 Protocol operation

Change NOTE 4 in 10.6 as shown:

NOTE 4—In the face of changing inputs, arbitrary loss, delay, reordering, and unsignaled interruption in participation, no protocol will meet its objectives. What can be said is that the objectives will be met after a known period of operation within specified limits, and what failures are most likely otherwise. MRP favors ensuring that registrations are made, at the expense of tolerating prolonged registrations. Thus, the LeaveAll mechanism, when it has an effect, most often implements "garbage collection." At the same time it will also ensure that failed registrations are (re-)established. These objectives cannot be ensured if the LeaveAllTime is set to zero; therefore, it is recommended to set the LeaveAllTime to a nonzero value as soon as is practicable.

10.7 Protocol specification

10.7.4 Protocol timers

10.7.4.3 leavealltimer

Change 10.7.4.3 as shown:

The Leave All Period Timer, leavealltimer, controls the frequency with which the LeaveAll state machine generates LeaveAll PDUs. The timer is required on a per-Port, per-MRP Participant basis. If LeaveAllTime is zero, the Leave All Period Timer is not started; otherwise, t~~T~~he Leave All Period Timer is set to a random value, T, in the range LeaveAllTime < T < 1.5 × LeaveAllTime when it is started. LeaveAllTime is defined in Table 10-7.

10.7.6 Protocol Action definitions

10.7.6.10 Start leavealltimer

Change 10.7.6.10 as shown:

When LeaveAllTime is nonzero, this c~~C~~auses leavealltimer to be started, in accordance with the definition of the timer in 10.7.4.3.

10.7.8 Registrar state machine

Change Table 10-4 as shown:

Table 10-4—Registrar state table

		STATE		
		IN	**LV**	**MT**
EVENT	**Begin!**	MT	MT	MT
	rNew!	New IN	New Stop leavetimer IN	New IN
	rJoinIn! || rJoinMt!	IN	Stop leavetimer IN	Join IN
	rLv! || rLA! || txLA! || Re-declare!	Start leave-timer[a] LV	-x-	-x-
	Flush!	Lv MT	Lv MT	MT
	leavetimer!	-x-	Lv MT	MT

[a] If operPointToPointMAC is TRUE, proceed directly to the Lv/MT transition as defined in the LV State for the leavetimer! Event. This avoids the delay resulting from the leavetimer. If operPointToPointMAC is FALSE, the state machine is executed as shown.

10.7.9 LeaveAll state machine

Change the text of 10.7.9 as shown (Table 10-5 remains unchanged):

A single LeaveAll state machine exists for each full MRP Participant. Leave All messages generated by this state machine also generate LeaveAll events against all the Applicant and Registrar state machines associated with that Participant and Port; hence, LeaveAll generation is treated by those state machines in the same way as reception of a LeaveAll message from an external source.

If the LeaveAllTime is changed from a value of zero to a nonzero value, a Begin! event shall be signaled to the LeaveAll state machine to cause it to reinitialize.

The detailed operation of this state machine is described in Table 10-5.

10.7.11 Timer values

Change the title of Table 10-7 as shown:

Table 10-7—MRP timer parameter default values

Insert the following subclause (10.7.14) after 10.7.13:

10.7.14 External control

The MRP External Control feature (12.32.4) specifies a managed object that the network manager uses to

a) Disable MRP attribute propagation (MAP) for the MRP Participant of a Bridge Port.
b) Read MRP attribute registrations that the MRP Participant receives.
c) Write MRP attribute values for the MRP Participant to declare.

There is one MRP External Control managed object for each MRP Participant (per Port per MRP application per MAP Context, as specified in 10.2 and 10.3.1).

Under default operation, the MRP External Control feature (externalControl attribute FALSE) is ignored by the operation of MRP and has no effect on MRP Attribute Propagation (MAP) specified in 10.3.

When the MRP External Control feature is enabled by network management (externalControl attribute TRUE), the operation of MAP in the Bridge performs as specified in 12.32.4, but all other aspects of MRP operation are performed as specified in Clause 10.

10.8 Structure and encoding of Multiple Registration Protocol Data Units (MRPDUs)

10.8.3 Packing and parsing MRPDUs

10.8.3.5 Handling of protocol versions

Change list item c) in the lettered list in 10.8.3.5 as shown:

c) Where the MRPDU carries a higher protocol version than the version implemented, then
   1) If a Message is encountered in which the AttributeType is not recognized, then that Message is discarded. This is achieved by discarding the successive VectorAttributes in the AttributeList until either an EndMark or the end of the PDU is reached. If an EndMark is reached, processing continues with the next Message.
   2) If a VectorAttribute is encountered in which the AttributeEvent is not recognized for the AttributeType concerned, then the VectorAttribute is discarded and processing continues with the next VectorAttribute, if the end of the PDU has not been reached.
   3) For the MSRP application (Clause 35), if a FirstValue is encountered in which a TLV is not recognized for the Type concerned, then the TLV is discarded and processing continues with the next TLV, if the end of the PDU has not been reached.

12. Bridge management

12.20 Management entities for FQTSS

Change the second paragraph of 12.20 as shown:

This managed resource comprises the following objects:

a) The Bandwidth Availability Parameter Table (12.20.1)
b) The Transmission Selection Algorithm Table (12.20.2)
c) The Priority Regeneration Override Table (12.20.3)
d) The SR Class to Priority Mapping Table (12.20.4)

12.20.1 Bandwidth Availability Parameter Table

Change 12.20.1, including Table 12-4, as shown:

There is one Bandwidth Availability Parameter Table per Port of a Bridge component. Each table row contains a set of parameters for each traffic class ~~that supports the credit-based shaper algorithm (8.6.8.2)~~ configured for use with time-sensitive streams, as detailed in Table 12-4. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of ~~p~~Ports and components.

Table 12-4—Bandwidth Availability Parameter Table row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
~~Traffic class~~trafficClass[c]	unsigned integer [0..7]	R	BE	~~34.3~~ 3.268, 8.6.8
deltaBandwidth	percentage	RW	BE	34.3
adminIdleSlope	unsigned integer	RW	BEC	34.3
operIdleSlope	unsigned integer	R	BE	34.3
classMeasurementInterval	unsigned integer	RW	be	34.3, 34.4, 34.6
lockClassBandwidth	Boolean	RW	be	34.3

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component support of FQTSS; E = Required for end station support of FQTSS~~.~~; b = Optional for Bridge or Bridge component support of FQTSS; e = Optional for end station support of FQTSS; C = Required for Bridge or Bridge component support of Stream reservation remote management.
[c] The term “trafficClass” in this table is the same as the “traffic class” definition of this standard.

The classMeasurementInterval attribute provides management control of the classMeasurementInterval parameter specified in 34.3. The classMeasurementInterval attribute uses units of nanoseconds, converted to/from units of seconds for use in 34.3. If management of classMeasurementInterval is not supported, the default values (34.5) are used as the fixed Port configuration.

If management of lockClassBandwidth is not supported, the default value of false is used as the fixed Port configuration.

12.20.2 Transmission Selection Algorithm Table

Change the text of 12.20.2 as shown (Table 12-5 remains unchanged):

There is one Transmission Selection Algorithm Table per Port of a Bridge component. ~~This is in addition to the Traffic Class Table managed object (12.6.3) that would also exist per Port of a bridge component.~~ Each table row contains a set of parameters for each traffic class that the Port supports, as detailed in Table 12-5.

NOTE—In order to manage the mapping of traffic class to priority, refer to the Traffic Class Table managed object (12.6.3).

12.20.3 Priority Regeneration Override Table

Change the text of 12.20.3 as shown (Table 12-6 remains unchanged):

There is one Priority Regeneration Override Table per Port of a Bridge component. This is in addition to the Priority Handling managed object (12.6.2) that would also exist per Port of a Bridge component. Each table row contains a set of parameters for each received priority value that is associated with an SR class (3.259, 6.9.4), as detailed in Table 12-6.

Insert the following subclause (12.20.4, including Table 12-7) after 12.20.3, and renumber the subsequent tables in Clause 12 accordingly:

12.20.4 SR Class to Priority Mapping Table

There is one SR Class to Priority Mapping Table per Bridge component. Each table row corresponds to an SR class, as detailed in Table 12-7. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of Ports and components, using Priority to specify the row to create/delete.

Table 12-7—SR Class to Priority Mapping Table row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
Priority	unsigned integer [0..7]	R	b	6.9.3
SRclassID	unsigned integer [0..6]	RW	b	35.2.2.9.2

[a] R = Read only access; RW = Read/Write access.
[b] b = Optional for Bridge or Bridge component support of FQTSS.

The SRclassID attribute specifies the SR class ID from Table 35-7 of 35.2.2.9.2, mapped to the associated Priority (SRclassPriority of 35.2.2.9.3). The default values for this managed object use the default values specified in 34.5 (i.e., Priority 3 for SRclassID 6 and Priority 2 for SRclassID 5).

If this managed object is not supported, the default values specified in 34.5 are used as the fixed configuration.

NOTE 1—This managed object is not needed for an end station, since according to the requirements of 35.2.2.9.3, an end station uses the SR class to priority mapping that the neighboring Bridge provides in SRP's Domain attribute.

NOTE 2—In order to manage the mapping of traffic class to priority, refer to the Traffic Class Table managed object (12.6.3).

NOTE 3—The maximum number of rows supported for this table is provided in the maxSRclasses parameter of 12.22.1.

12.22 Stream Reservation Protocol (SRP) entities

12.22.1 SRP Bridge Base Table

Change Table 12-14 (formerly Table 12-13) as shown:

Table 12-14—SRP Bridge Base Table row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
msrpEnabledStatus	Boolean	RW	B	35.2.1.4 item d)
talkerPruning	Boolean	RW	B	35.2.1.4 item b)
msrpMaxFanInPorts	unsigned integer	RW	B	35.2.1.4 item f)
msrpLatencyMaxFrameSize	unsigned integer	RW	B	35.2.1.4 item g)
talkerVlanPruning	Boolean	RW	b	35.2.1.4 item l)
maxSRclasses	unsigned integer	R	b	35.2.1.4 item m)

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component support of SRP; E = Required for end station support of SRP; b = Optional for Bridge or Bridge component support of SRP.

12.22.2 SRP Bridge Port Table

Change 12.22.2, including Table 12-15 (formerly Table 12-14) as shown:

There is one SRP Configuration Parameter Table per Port of a Bridge component. Each table row contains a set of parameters for each MSRP entity per ~~p~~Port, as detailed in Table 12-15.

Table 12-15—SRP Bridge Port Table row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
msrpPortEnabledStatus	Boolean	RW	B	35.2.1.4 item e)
MSRP Failed Registrations	counter	R	B	10.7.12.1
MSRP Last PDU Origin	MAC Address	R	B	10.7.12.2
SR_PVID	unsigned integer[1..4094]	RW	B	Table 9-2, 35.2.1.4 item i)
talkerPruningPerPort	Boolean	RW	b	35.2.1.4 item k)

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component support of SRP; E = Required for end station support of SRP; b = Optional for Bridge or Bridge component support of SRP.

Insert the following subclauses (12.22.6 and 12.22.7, including Table 12-19 and Table 12-20) after 12.22.5:

12.22.6 SRP Stream Preload Table

There is one SRP Stream Preload Table per Bridge component. Each table contains a set of parameters for each StreamID that is preloaded on the Bridge as it initializes (i.e., powers up/reboots), as detailed in Table 12-19. The preloading of Table 12-19 on each Port of the Bridge is specified in 12.22.7. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of Ports and components.

Table 12-19—SRP Stream Preload Table row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
StreamID	octet string(size(8))	RW	b	35.2.2.8.2
Stream Destination Address	MAC Address	RW	b	35.2.2.8.3 item a)
Stream VLAN ID	unsigned integer [0..4094]	RW	b	35.2.2.8.3 item b)
MaxFrameSize	unsigned integer [0..65535]	RW	b	35.2.2.8.4 item a)
MaxIntervalFrames	unsigned integer [0..65535]	RW	b	35.2.2.8.4 item b)
Data Frame Priority	unsigned integer [0..7]	RW	b	35.2.2.8.5 item a)
Rank	unsigned integer [0..1]	RW	b	35.2.2.8.5 item b)

[a] R = Read only access; RW = Read/Write access.
[b] b = Optional for Bridge or Bridge component support of SRP.

12.22.7 SRP Reservations Preload Table

There is one SRP Reservations Preload Table per reservation direction per Port of a Bridge component. Each table contains a set of parameters for each Talker or Listener registration that is preloaded on the Bridge as it initializes (i.e., powers up/reboots), as detailed in Table 12-20. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of Ports and components.

Table 12-20—SRP Reservations Preload Table row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
StreamID	octet string(size(8))	RW	b	35.2.2.8.2
Direction	unsigned integer[0..1]	RW	b	35.2.1.2
Accumulated Latency	unsigned integer	RW	b	35.2.2.8.6

[a]R = Read only access; RW = Read/Write access.
[b]b = optional for Bridge or Bridge component support of SRP.

The SRP preload tables (Table 12-19 and Table 12-20) are used to initialize Streams within each Bridge as it powers up, to preload the Stream registrations that will later be provided by operation of SRP. When the Bridge initializes, the SRP preload tables shall be used for a two phase operation as follows:

In the first phase, the Bridge iterates Table 12-20 to find rows for which the Direction is Talker. For each Talker row, if the StreamID in Table 12-20 is also found in Table 12-19, the Bridge shall

a) Invoke the MVRP ES_REGISTER_VLAN_MEMBER service primitive (11.2.3.2.1) on the Port. The attribute_type parameter carries the VID Vector Attribute Type (11.2.3.1.6). The attribute_value parameter carries the Stream VLAN ID for the associated StreamID in Table 12-19.
b) Invoke MSRP REGISTER_STREAM.request (35.2.3.1.1) on the Port. The attribute_type parameter carries the Talker Advertise Vector Attribute Type [item a) in 35.2.2.4]. The Declaration Type in the attribute_value parameter is Talker Advertise. The attribute_value parameter carries the following parameters from Table 12-19: StreamID, DataFrameParameters.destination_address from Stream Destination Address, DataFrameParameters.vlan_identifier from Stream VLAN ID, TSpec.MaxFrameSize, TSpec.MaxIntervalFrames, PriorityAndRank.Data Frame Priority, and PriorityAndRank.Rank. The attribute_value parameter carries the following per-Port parameters from Table 12-20: Accumulated Latency. The Failure Information in the attribute_value parameter is not applicable to Talker Advertise.

In the second phase, the Bridge iterates Table 12-20 to find rows for which the Direction is Listener. For each Listener row, if the StreamID was processed as a Talker during the first phase, then the Bridge shall:

c) Invoke MSRP REGISTER_ATTACH.request (35.2.3.1.5) on the Port. The attribute_type parameter carries the Listener Vector Attribute Type [item c) in 35.2.2.4]. The Declaration Type in the attribute_value parameter is Listener Ready. The attribute_value parameter carries StreamID from Table 12-20.
d) Invoke the MVRP ES_REGISTER_VLAN_MEMBER service primitive (11.2.3.2.1) on the Port. The attribute_type parameter carries the VID Vector Attribute Type (11.2.3.1.6). The attribute_value parameter carries the Stream VLAN ID for the associated StreamID in Table 12-19.
e) Update the Dynamic Reservation Entries (8.8.7) for the Port to Forwarding, using the Stream Destination Address for the associated StreamID in Table 12-19.
f) For traffic class *N* that is mapped to the Data Frame Priority of Table 12-19, increase the *operIdleSlope(N)* variable [item d) in 34.3] using the bandwidth requirements of the Stream (MaxFrameSize and MaxIntervalFrames of Table 12-19).

Insert the following subclause (12.32, including Table 12-38 through Table 12-41) after 12.31.4:

12.32 Stream reservation remote management

Clause 46 defines Time-Sensitive Networking (TSN) configuration, including a Centralized Network Configuration (CNC) station that uses managed objects for configuration of each Bridge. This subclause specifies managed objects within the Bridge that can be used by the TSN CNC station or any other management component.

This managed resource comprises the following objects:

a) Bridge Delay (12.32.1)
b) Propagation Delay (12.32.2)
c) Static Trees (12.32.3)
d) MRP External Control (12.32.4)

12.32.1 Bridge Delay

This subclause specifies attributes of a managed object that is used to determine the delay of frames as they pass through the Bridge's relay. There is one Bridge Delay managed object per Port pair per traffic class of a Bridge component. Each set of Bridge Delay attributes is accessed using three indices: ingress Port, egress Port, and traffic class. The set of managed object attributes is shown in Table 12-38.

Table 12-38—Bridge Delay attributes

Name	Data type	Operations supported[a]	Conformance[b]	References
independentDelayMin	unsigned integer	R	B	12.32.1.1
independentDelayMax	unsigned integer	R	B	12.32.1.1
dependentDelayMin	unsigned integer	R	B	12.32.1.2
dependentDelayMax	unsigned integer	R	B	12.32.1.2

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component support of Stream reservation remote management; b = Optional for Bridge or Bridge component support of Stream reservation remote management.

For each set of Bridge Delay attributes, the total delay through the Bridge consists of the delay that is independent of frame length, plus the delay that is dependent on frame length.

Each delay is provided as a range, with both minimum attribute and maximum attribute.

The delays represent the worst-case range per the design of the Bridge, and are not measured. If the Bridge design varies based on the configuration of features in the Bridge, the delays are returned according to the current configuration of the Bridge.

NOTE 1—In order to obtain the most accurate delays, the TSN CNC typically configures the Stream from Talker to Listeners, and then reads the delays for the Ports that ingress/egress that Stream. For example, use of IEEE Std 802.1CB could change the implementation in the Bridge, and therefore it is best to configure those features in the Bridge prior to reading the delays.

Each delay is provided as if the corresponding frame encounters zero delay for transmission selection (8.6.8). This occurs when the queue for the frame's traffic class is empty, the frame's traffic class has permission to transmit, and the egress Port is idle (not transmitting).

NOTE 2—Computation of delay factors due to use of multiple traffic classes, frame preemption, traffic shaping, and traffic scheduling is the responsibility of the TSN CNC, and therefore these factors are not included in the delay attributes of this managed object. For example, assume that a frame of a TSN Stream encounters 12 µs of delay due to traffic scheduling (i.e., waiting for gate to open), 5 µs due to a previous frame of the same traffic class, and 700 ns to 800 ns of hardware delay (e.g., PHYs and relay). For this example, independentDelayMin is 700, and independentDelayMax is 800 (i.e., the extra 17 µs for queuing/scheduling is not included in this managed object).

12.32.1.1 independentDelayMin/Max

These attributes provide the portion of delay that is independent of frame length.

Each length-independent delay attribute is an unsigned integer in units of nanoseconds.

The independentDelayMin attribute provides the minimum length-independent delay for a frame to forward from ingress Port to egress Port for this traffic class. If the actual delay is a fraction of a nanosecond, independentDelayMin shall be the greatest integer number of nanoseconds that is less than or equal to the actual delay.

The independentDelayMax attribute provides the maximum length-independent delay for a frame to forward from ingress Port to egress Port for this traffic class. If the actual delay is a fraction of a nanosecond, independentDelayMax shall be the least integer number of nanoseconds that is greater than or equal to the actual delay.

NOTE 1—Since the minimum to maximum range is intended to be worst-case, the rules for handling a fraction of a nanosecond are intended to result in the widest range.

The delay begins when the message timestamp point of the ingress frame passes the reference plane marking the boundary between the network media and PHY. The delay ends when the message timestamp point of the egress frame passes the reference plane marking the boundary between the network media and PHY. The message timestamp point is specified by IEEE Std 802.1AS for various media, near the start of the frame.

NOTE 2—This delay includes all aspects of length-independent delay for a frame that is forwarded, including handling of error conditions.

12.32.1.2 dependentDelayMin/Max

These attributes provide the portion of delay that is dependent on frame length, where frame length is the number of octets that transfer across the MAC Service interfaces. Each length-dependent delay attribute specifies the time for a single octet of the frame to transfer from ingress to egress.

Each length-dependent delay attribute is an unsigned integer in units of picoseconds.

The dependentDelayMin attribute provides the minimum length-dependent delay from ingress Port to egress Port for this traffic class. If the actual delay is a fraction of a picosecond, dependentDelayMin shall be the greatest integer number of picoseconds that is less than or equal to the actual delay.

The dependentDelayMax attribute provides the maximum length-dependent delay from ingress Port to egress Port for this traffic class. If the actual delay is a fraction of a picosecond, dependentDelayMax shall be the least integer number of picoseconds that is greater than or equal to the actual delay.

NOTE—The length-dependent delay typically includes the time to receive and store each octet of the frame, and this time depends on the link speed of the ingress Port. For example, if the ingress Port is operating at 1 Gb/s, both dependentDelayMin and dependentDelayMax can return 8000. If an internal communication mechanism exists within the Bridge's architecture (i.e., in the relay), the time for that mechanism is added to the frame-dependent delay. For example, if a 100 Mb/s serial link connects the ingress and egress Ports, the previous example with 1 Gb/s link speed can return 88000 for both dependentDelayMin and dependentDelayMax.

12.32.2 Propagation Delay

This subclause specifies attributes of a managed object used to determine the delay along the network media (e.g., cable) for a frame transmitted from the specified Port of this station to the neighboring Port on a different station. There is one Propagation Delay managed object per Port of a station (i.e., Bridge or end station). The set of managed object attributes is shown in Table 12-39.

Table 12-39—Propagation Delay attributes

Name	Data type	Operations supported[a]	Conformance[b]	References
txPropagationDelay	unsigned integer	R	BE	12.32.2.1

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component support of Stream reservation remote management; E = Required for end station support of Stream reservation remote management.

12.32.2.1 txPropagationDelay

The txPropagationDelay begins when the message timestamp point of an egress frame passes the reference plane marking the boundary between the network media and PHY of this station's Port. The txPropagationDelay ends when the message timestamp point of an ingress frame on the neighboring station's Port passes the reference plane marking the boundary between the network media and PHY. The message timestamp point is specified by IEEE Std 802.1AS for various media, near the start of the frame.

The txPropagationDelay attribute is an unsigned integer in units of nanoseconds. If the actual propagation delay is measured to a fraction of a nanosecond, the txPropagationDelay value shall be least integer number of nanoseconds that is greater than or equal to the actual delay.

NOTE 1—Propagation delay in one direction of transmit is not necessarily the same as propagation delay of transmit in the reverse direction. In order to determine if a delay asymmetry exists, management can read the txPropagationDelay attribute on the neighboring station's Port. The txPropagationDelay attribute of the neighboring station's Port effectively provides the propagation delay for receiving frames on this station's Port.

NOTE 2—The txPropagationDelay attribute is typically measured using a time synchronization protocol. For example, when IEEE Std 802.1AS is in use, the value of this attribute can be obtained using the managed objects neighborPropDelay and delayAsymmetry.

NOTE 3—One of the primary purposes of the txPropagationDelay attribute is to add to Bridge Delay in order to compute total latency for a Stream. Although time synchronization protocols based on IEEE Std 1588 typically measure propagation delay to units of 2^{-16} ns, that degree of precision is not required for txPropagationDelay, since Bridge Delay is expressed as nanoseconds.

12.32.3 Static Trees

This subclause specifies attributes of a managed object used to determine if the static trees feature is supported by the Bridge. There is one Static Trees managed object that applies to the Bridge. The set of managed object attributes is shown in Table 12-40.

Table 12-40—Static Trees attributes

Name	Data type	Operations supported[a]	Conformance[b]	References
staticTreesSupported	Boolean	R	B	12.32.3.1

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component support of Stream reservation remote management.

12.32.3.1 staticTreesSupported

For some use cases, the TSN CNC needs to configure static (persistent) trees through Bridges from Talker to Listeners (e.g., for use of IEEE Std 802.1CB). The Traffic Engineering Multiple Spanning Tree Instance Identifier (TE-MSTID) can be leveraged for these use cases. A VLAN ID is allocated to the TE-MSTID in the MST Configuration Table if it is not under control of either a spanning tree protocol or IS-IS. Static Filtering Entries (8.8.1) specify frame forwarding and filtering for VLAN IDs allocated to the TE-MSTID.

The TE-MSTID feature is used as the basis of a set of features called Provider Backbone Bridge Traffic Engineering (PBB-TE) as specified in 25.10. For many TSN CNCs, use of the TE-MSTID feature is essential, but the additional features of PBB-TE are not necessary. The TSN CNC uses the staticTreesSupported attribute to determine that TE-MSTID is supported by the Bridge. A Bridge that supports PBB-TE without support of Stream reservation remote management is not required to implement the staticTreesSupported attribute.

A Bridge that supports Stream reservation remote management shall return the value TRUE from the staticTreesSupported attribute, and shall support the TE-MSTID as specified in 5.5.2.

NOTE—In order to determine support for TE-MSTID by the Bridge, the TSN CNC attempts to read the staticTreesSupported attribute. If the read operation returns an error, or if the read returns FALSE, then the TSN CNC concludes that TE-MSTID is not supported.

12.32.4 MRP External Control

In the Centralized network/distributed user model shown in Figure 46-2, user configuration data is exchanged in a distributed manner by each Talker/Listener end station, and the network is configured in a centralized manner (using a CNC). A Bridge near the Talker/Listener acts as a proxy with the CNC. One protocol solution for this model is to use the MRP protocol for user data (i.e., MSRP, MVRP, and/or MMRP applications), and a network management protocol as the proxy with the network manager (i.e., CNC). This enables the end stations to use the same MRP protocol as the Fully distributed model of Figure 46-1.

This subclause specifies attributes of a managed object that the network manager uses to

a) Disable MRP attribute propagation (MAP) for the MRP Participant of a Bridge Port.
b) Read MRP attribute registrations that the MRP Participant receives.
c) Write MRP attribute values for the MRP Participant to declare.

There is one MRP External Control managed object for each MRP Participant (per Port per MRP application per MAP Context, as specified in 10.2 and 10.3.1).

For the MSRP application, although 35.2.4.5 implies that a single MAP Context exists, the specifications of 35.1.3.1 ensure that every distinct value of vlan_identifier propagates along the corresponding VLAN context (10.3.1). Therefore, for the purposes of indexing this managed object, MSRP uses the vlan_identifier as the MAP Context identifier.

The set of managed object attributes is shown in Table 12-41.

Table 12-41—MRP External Control attributes

Name	Data type	Operations supported[a]	Conformance[b]	References
externalControl	Boolean	RW	B	12.32.4.1
indicationList	octet string	R	B	12.32.4.2
indicationListLength	unsigned integer	R	B	12.32.4.3
indicationChangeCounter	counter	R	B	12.32.4.4
adminRequestList	octet string	RW	B	12.32.4.5
adminRequestListLength	unsigned integer	RW	B	12.32.4.6
operRequestList	octet string	R	B	12.32.4.7
operRequestListLength	unsigned integer	R	B	12.32.4.8

[a] R = Read only access; RW = Read/Write access.
[b] B = Required for Bridge or Bridge component when both Stream reservation remote management and the corresponding MRP application are supported.

12.32.4.1 externalControl

When the externalControl attribute is FALSE, MRP attributes propagate on the MRP Participant according to the specifications for MAP in 10.3 and specifications of the MRP Application (e.g., 35.2.4). Ports with the externalControl attribute FALSE are considered as candidates for the MRP Application's MAP Context. The remaining attributes of this managed object (12.32.4.2 through 12.32.4.8) are ignored by Ports with the externalControl attribute FALSE.

When the externalControl attribute is TRUE, the MRP Participant is removed from the MRP Application's MAP Context. The MRP Participant performs all other aspects of MRP, including MRP operation (10.6), MRP specifications (10.7), and MRPDU encodings (10.8). The remaining attributes of this managed object (12.32.4.2 through 12.32.4.8) act as an application component as specified in 10.2. The application component stores MAD indications for registration received on the Port, and invokes MAD requests for declarations on the Port.

The default value of the externalControl attribute is FALSE.

12.32.4.2 indicationList

As specified in 10.2, the MAD_Join.indication primitive is invoked in the application component when MRP detects that an MRP attribute has joined (i.e., registered), and the MAD_Leave.indication primitive is invoked in the application component when MRP detects that the MRP attribute has left (i.e., no longer registered). When the externalControl attribute is TRUE, the indicationList attribute stores the list of all joined MRP attributes for the MRP Participant.

When the MAD_Join.indication primitive is invoked, the MRP Participant forms a sequence of octets, *temporaryIndication*, from the MAD_Join.indication parameters as follows:

a) The first octet of *temporaryIndication* contains the *attribute_type* parameter.
b) The second octet of *temporaryIndication* contains the length of the *attribute_value* parameter as a single octet.

c) Subsequent octets of *temporaryIndication* contain the sequence of octets for the *attribute_value* parameter.

NOTE 1—The *new* parameter is not used for indicationList.

The MRP Participant searches the indicationList to determine if the same octets as *temporaryIndication* (i.e., same *attribute_type* and *attribute_value*) exist. If no match is found, the contents of *temporaryIndication* are appended to the end of indicationList. The result of this specification is that the indicationList consists of an octet string that represents multiple MAD_Join.indication primitives in the order initially invoked.

When the MAD_Leave.indication primitive is invoked, the MRP Participant forms a sequence of octets, *temporaryIndication*, as specified for MAD_Join.indication. The MRP Participant searches the indicationList to determine if the same octets as *temporaryIndication* exist. If a match is found, the attribute's octets (i.e., matching *temporaryIndication)* are removed from the indicationList.

NOTE 2—When the indication's *attribute_type* is Listener Vector Attribute Type, the Listener's Declaration Type (35.2.1.3) is encoded as the last octet of *attribute_value*, even though the Declaration Type is not encoded in the MRP FirstValue (10.8.2.7), but the MRP FourPackedEvent (10.8.2.10.2). For the Listener Enhanced Vector Attribute Type, the Listener's Declaration Type is not encoded as the last octet, since the equivalent information is available in the ListenerStatus element (46.2.5.1).

The default value of the indicationList attribute is the empty octet string.

NOTE 3—The attribute encoding of indicationList does not correspond directly to the MRPDU. For example, if the received MRPDU uses NumberOfValues of three, then the MRPDU encodes only one attribute value (FirstValue), whereas indicationList encodes three distinct attribute type/length/value triplets.

When the externalControl attribute is FALSE, the indicationList attribute is ignored by the MRP Participant, and returns the empty octet string.

12.32.4.3 indicationListLength

The indicationListLength attribute returns the number of octets in the indicationList attribute.

When the externalControl attribute is FALSE, the indicationListLength attribute returns zero.

12.32.4.4 indicationChangeCounter

The indicationChangeCounter attribute increments by one for each change to indicationList as specified in 12.32.4.2.

The default value of the indicationChangeCounter attribute is zero.

When the externalControl attribute is FALSE, the indicationChangeCounter attribute returns zero.

12.32.4.5 adminRequestList

The adminRequestList attribute provides the administrative value of the current list of MAD requests for the MRP Participant (operRequestList). Changes to adminRequestList result in invocation of MAD_Join.request and MAD_Leave.request primitives (12.32.4.7). Each attribute in adminRequestList is encoded as the *attribute_type* parameter as a single octet, followed by the length of the *attribute_value* parameter as a single octet, followed by a sequence of octets for the *attribute_value* parameter.

NOTE—The *new* parameter is not used for adminRequestList.

The default value of the adminRequestList attribute is the empty octet string.

When the externalControl attribute is TRUE, the adminRequestList attribute is copied to the operRequestList attribute as soon as possible according to the implementation. For example, the implementation could copy the list between each MRPDU transmit, or it could copy the list after all attributes in the previous operRequestList complete transmission as MRPDUs.

When the externalControl attribute is FALSE, the adminRequestList attribute is ignored by the MRP Participant, but it retains its value.

12.32.4.6 adminRequestListLength

The adminRequestListLength attribute provides the administrative value for the number of octets in adminRequestList.

The default value of the adminRequestListLength attribute is zero.

When the externalControl attribute is TRUE, the adminRequestListLength attribute is copied to the operRequestListLength attribute at the same time that the adminRequestList attribute is copied to the operRequestList attribute.

When the externalControl attribute is FALSE, the adminRequestListLength attribute is ignored by the MRP Participant, but it retains its value.

12.32.4.7 operRequestList

The operRequestList attribute is the operational value of the current list of MAD requests for the MRP Participant. Prior to copying adminRequestList to operRequestList, the lists are compared as follows:

a) If an attribute exists in adminRequestList that does not exist in operRequestList, invoke MAD_Join.request for that attribute, with the *new* parameter set TRUE.
b) If an attribute exists in operRequestList that does not exist in adminRequestList, invoke MAD_Leave.request for that attribute.

When the comparison is complete, adminRequestList is copied to operRequestList.

The default value of the operRequestList attribute is the empty octet string.

12.32.4.8 operRequestListLength

The operRequestListLength attribute is the operational value of the adminRequestListLength attribute, and it is copied at the same time that adminRequestList attribute is copied to operRequestList.

The default value of the operRequestListLength attribute is zero.

17. Management Information Base (MIB)

17.4 Security considerations

17.4.12 Security considerations of the IEEE8021-FQTSS-MIB

Change the third paragraph and lettered list in 17.4.12 as shown:

The following tables and objects in the IEEE8021-FQTSS-MIB can be manipulated to interfere with the operation of the forwarding and queuing mechanisms in a manner that would be detrimental to the transmission of time-sensitive streams:

ieee8021FqtssDeltaBandwidth
ieee8021FqtssAdminIdleSlopeMs
ieee8021FqtssAdminIdleSlopeLs
ieee8021FqtssClassMeasurementInterval
ieee8021FqtssSrClassId
ieee8021FqtssTxSelectionAlgorithmID
ieee8021FqtssPriorityRegenOverride

a) ieee8021FqtssDeltaBandwidth can be manipulated to reduce the amount of bandwidth available to a given SR class.
b) ieee8021FqtssAdminIdleSlopeMs and ieee8021FqtssAdminIdleSlopeLs can be manipulated to change the overall amount of bandwidth available to stream traffic on a Port, in a network where SRP is not used for stream reservations.
c) ieee8021FqtssClassMeasurementInterval can be manipulated to change the measurement interval that is used, together with TSpec, to calculate reserved bandwidth.
d) ieee8021FqtssSrClassId can be manipulated to change the priority mapping of a SRclassID to a priority. This can impact the behavior of Streams.
e) ~~c)~~ ieee8021FqtssTxSelectionAlgorithmID can be manipulated in order to apply the wrong transmission selection algorithm to a traffic class that was being used by stream traffic.
f) ~~d)~~ ieee8021FqtssPriorityRegenOverride can be manipulated to disrupt stream traffic within an SRP domain with non-stream traffic entering from outside the SRP domain boundary.

17.4.14 Security considerations of the IEEE8021-SRP MIB

Change the third paragraph and lettered list in 17.4.14 as shown:

The following tables and objects in the IEEE8021-SRP MIB can be manipulated to interfere with the operation of the stream reservation mechanisms in a manner that would be detrimental to the transmission of the associated stream data:

ieee8021SrpBridgeBaseMsrpEnabledStatus
ieee8021SrpBridgeBaseMsrpTalkerPruning
ieee8021SrpBridgeBaseMsrpMaxFanInPorts
ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize
ieee8021SrpBridgeBaseMsrpTalkerVlanPruning
ieee8021SrpBridgePortMsrpEnabledStatus
ieee8021SrpBridgePortSrPvid
ieee8021SrpBridgeBaseMsrpTalkerPruningPerPort

a) ieee8021SrpBridgeBaseMsrpEnabledStatus can be manipulated to enable or disable MSRP operations for the entire Bridge.

b) ieee8021SrpBridgeBaseMsrpTalkerPruning can be manipulated to stop the propagation of Talker attributes if Listeners are not configured to support Talker Pruning.

c) ieee8021SrpBridgeBaseMsrpMaxFanInPorts can be manipulated to set the number of ingress ~~p~~Ports supporting streaming down to one, which would stop Talkers streams from coming in on any other ~~p~~Port.

d) ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize can be manipulates to set a frame size that is so large or so small that it causes the Bridge to calculate unreasonable maximum latency.

e) ieee8021SrpBridgeBaseMsrpTalkerVlanPruning can be manipulated to change the behavior of a Bridge regarding Talker declarations. This can either restrict Talker declarations in an unintended way or cause Talker declarations to be forwarded on unintended Ports.

f) ~~e)~~ ieee8021SrpBridgePortMsrpEnabledStatus can be manipulated to enable or disable MSRP on a particular ~~p~~Port, perhaps allowing sensitive stream data to be sent to unacceptable devices.

g) ieee8021SrpBridgePortSrPvid can be manipulated to move Streams to a VLAN that has been blocked by management, thus disabling reception of the Stream by one or more Listeners.

h) ieee8021SrpBridgeBaseMsrpTalkerPruningPerPort can be manipulated to disable MAC address pruning per Port, causing Talker declarations to be forwarded on unintended Ports.

17.7.12 Definitions for the IEEE8021-FQTSS-MIB module

Replace 17.7.12 with the following text:

```
IEEE8021-FQTSS-MIB DEFINITIONS ::= BEGIN

-- ============================================================
-- MIB for support of the Forwarding & Queuing Enhancements
-- for Time Sensitive Streams (FQTSS) in IEEE 802.1Q Bridges.
-- ============================================================

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION,
    TruthValue,
    RowStatus
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    ieee802dot1mibs,
    IEEE8021PriorityValue
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBaseComponentId,
    ieee8021BridgeBaseEntry,
    ieee8021BridgeBasePort,
    ieee8021BridgeBasePortEntry
        FROM IEEE8021-BRIDGE-MIB
    BridgeId
        FROM BRIDGE-MIB
    ;

ieee8021FqtssMib MODULE-IDENTITY
    LAST-UPDATED "201810040000Z" -- October 4, 2018
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "  WG-URL: http://ieee802.org/1/
         WG-EMail: STDS-802-1-L@IEEE.ORG

           Contact: IEEE 802.1 Working Group Chair
            Postal: C/O IEEE 802.1 Working Group
                    IEEE Standards Association
                    445 Hoes Lane
                    Piscataway
                    NJ 08854
                    USA
```

```
            E-mail: STDS-802-1-L@IEEE.ORG"
        DESCRIPTION
            "The Bridge MIB module for managing devices that support
            the Forwarding and Queuing Enhancements
            for Time Sensitive Streams.

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

            Copyright (C) IEEE (2018).
            This version of this MIB module is part of IEEE Std 802.1Q;
            see the draft itself for full legal notices."

        REVISION "201810040000Z" -- October 4, 2018
        DESCRIPTION
                "Published as part of IEEE 802.1Qcc-2018.
                      Added managed objects for Stream Reservation
                      Protocol (SRP) Enhancements and Performance
                      Improvements"

        REVISION "201806280000Z" -- June 28, 2018
        DESCRIPTION
                "Published as part of IEEE Std 802.1Q 2017.
                Cross references corrected and updated. "

        REVISION "201512020000Z" -- December 2, 2015
        DESCRIPTION
                "Published as part of IEEE Std 802.1Q 2014 Cor-1.
                ETS code point added to the textual convention
                IEEE8021FqtssTxSelectionAlgorithmIDValue "

        REVISION "201412150000Z" -- December 15, 2014
        DESCRIPTION
                "Published as part of IEEE Std 802.1Q 2014 revision.
                Cross references updated and corrected."

        REVISION     "201102270000Z" -- February 27, 2011
        DESCRIPTION
             "Minor edits to contact information etc. as part of
              2011 revision of IEEE Std 802.1Q."

        REVISION     "200910010000Z" -- October 1, 2009
        DESCRIPTION
             "Initial revision, included in IEEE 802.1Qav."
        ::= { ieee802dot1mibs 16 }

-- ================================================================
-- Textual Conventions
```

```
-- ============================================================

IEEE8021FqtssTrafficClassValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "An 802.1 FQTSS traffic class value.
        This is the numerical value associated with a traffic
        class in a Bridge. Larger values are associated with
        higher priority traffic classes."
    REFERENCE   "12.20.1"
    SYNTAX      Unsigned32 (0..7)

IEEE8021FqtssDeltaBandwidthValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "An 802.1 FQTSS delta bandwidth percentage,
        represented as a fixed point number scaled by
        1,000,000."
    REFERENCE   "12.20.1, 34.4"
    SYNTAX      Unsigned32 (0..100000000)

IEEE8021FqtssTxSelectionAlgorithmIDValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "An 802.1 transmission selection algorithm identifier
         value. This is an integer, with the following
         interpretation placed on the value:

         0: Strict priority algorithm,
         1: Credit-based shaper algorithm,
         2: Enhanced Transmission Selection algorithm,
         3-255: Reserved for future standardization,
         256-4294967295: Vendor-specific transmission selection
                         algorithm identifiers, consisting of a
                         four-octet integer, where the most
                         significant 3 octets hold an OUI or CID value,
                         and the least significant octet holds
                         an integer value in the range 0-255
                         assigned by the owner of the OUI or CID."
    REFERENCE   "8.6.8, 12.20.2"
    SYNTAX  Unsigned32

-- ============================================================
-- subtrees in the FQTSS MIB
-- ============================================================
```

```
ieee8021FqtssNotifications
    OBJECT IDENTIFIER ::= { ieee8021FqtssMib 0 }

ieee8021FqtssObjects
    OBJECT IDENTIFIER ::= { ieee8021FqtssMib 1 }

ieee8021FqtssConformance
    OBJECT IDENTIFIER ::= { ieee8021FqtssMib 2 }

ieee8021FqtssBap
    OBJECT IDENTIFIER ::= { ieee8021FqtssObjects 1 }

ieee8021FqtssMappings
    OBJECT IDENTIFIER ::= { ieee8021FqtssObjects 2 }

ieee8021FqtssBapX
    OBJECT IDENTIFIER ::= { ieee8021FqtssObjects 3 }

-- ============================================================
-- The ieee8021FqtssBap subtree
-- This subtree defines the objects necessary for the management
-- of bandwidth allocation for queues that support FQTSS.
-- ============================================================

-- ============================================================
-- the ieee8021FqtssBapTable
-- ============================================================

ieee8021FqtssBapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssBapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of bandwidth availability
        parameters for each traffic class that supports the
        credit-based shaper algorithm.
        All writable objects in this table must be
        persistent over power up restart/reboot."
    REFERENCE   "12.20.1"
    ::= { ieee8021FqtssBap 1 }

ieee8021FqtssBapEntry OBJECT-TYPE
    SYNTAX      Ieee8021FqtssBapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing bandwidth allocation
```

```
        information for each traffic class that supports the
        credit-based shaper algorithm. Rows in the table are
        automatically created and deleted as a result of the
        operation of the algorithm described in 34.5. "
    INDEX  { ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort,
             ieee8021FqtssBAPTrafficClass  }
    ::= { ieee8021FqtssBapTable 1 }

Ieee8021FqtssBapEntry ::=
    SEQUENCE {
        ieee8021FqtssBAPTrafficClass
            IEEE8021FqtssTrafficClassValue,
        ieee8021FqtssDeltaBandwidth
            IEEE8021FqtssDeltaBandwidthValue,
        ieee8021FqtssOperIdleSlopeMs
            Unsigned32,
        ieee8021FqtssOperIdleSlopeLs
            Unsigned32,
        ieee8021FqtssAdminIdleSlopeMs
            Unsigned32,
        ieee8021FqtssAdminIdleSlopeLs
            Unsigned32,
        ieee8021FqtssBapRowStatus
            RowStatus
        }

ieee8021FqtssBAPTrafficClass OBJECT-TYPE
    SYNTAX      IEEE8021FqtssTrafficClassValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The traffic class number associated with the row of
        the table.

        A row in this table is created for each traffic class
        that supports the credit-based shaper algorithm. The
        recommended mappings of priorities to traffic classes
        for support of the credit-based shaper algorithm are
        described in 34.5."
    REFERENCE   "12.20.2, 34.3, 34.5"
    ::= { ieee8021FqtssBapEntry 1 }


ieee8021FqtssDeltaBandwidth OBJECT-TYPE
    SYNTAX      IEEE8021FqtssDeltaBandwidthValue
    UNITS       "percent"
    MAX-ACCESS  read-write
```

```
        STATUS      current
        DESCRIPTION
            "The value of the deltaBandwidth parameter
            for the traffic class.
            This value is represented as a fixed point number
            scaled by a factor of 1,000,000; i.e., 100,000,000
            (the maximum value) represents 100%.

            The default value of the deltaBandwidth parameter
            for the highest numbered traffic class that supports
            the credit-based shaper algorithm is 75%; for all
            lower numbered traffic classes that support the
            credit-based shaper algorithm the default value is 0%.

            The value of this object MUST be retained across
            reinitializations of the management system."
        REFERENCE   "12.20.1, 34.3"
        ::= { ieee8021FqtssBapEntry 2}

    ieee8021FqtssOperIdleSlopeMs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "bits per second"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The most significant 32 bits of the bandwidth,
            in bits per second, that is currently allocated to the
            traffic class (idleSlope(N)). This object MUST be read
            at the same time as ieee8021FqtssOperIdleSlopeLs,
            which represents the LS 32 bits of the value, in order
            for the read operation to succeed.

            If SRP is supported and in operation, then the reserved
            bandwidth is determined by the operation of SRP; otherwise,
            the value of ieee8021FqtssOperIdleSlopeMs is equal to
            the value of ieee8021FqtssAdminIdleSlopeMs.

            The value of this object MUST be retained across
            reinitializations of the management system."
        REFERENCE   "12.20.1, 34.3"
        ::= { ieee8021FqtssBapEntry 3 }

    ieee8021FqtssOperIdleSlopeLs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "bits per second"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
        "The least significant 32 bits of the bandwidth,
        in bits per second, that is currently allocated to the
        traffic class (idleSlope(N)). This object MUST be read
        at the same time as ieee8021FqtssOperIdleSlopeMs,
        which represents the LS 32 bits of the value, in order
        for the read operation to succeed.

        If SRP is supported and in operation, then the reserved
        bandwidth is determined by the operation of SRP; otherwise,
        the value of ieee8021FqtssOperIdleSlopeLs is equal to
        the value of ieee8021FqtssAdminIdleSlopeMs.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "12.20.1, 34.3"
    ::= { ieee8021FqtssBapEntry 4 }

ieee8021FqtssAdminIdleSlopeMs OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "bits per second"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The most significant 32 bits of the bandwidth,
        in bits per second, that the manager desires to allocate
        to the traffic class as  idleSlope(N). This object MUST be
        read or written at the same time as
        ieee8021FqtssAdminIdleSlopeLs,
        which represents the LS 32 bits of the value, in order
        for the read or write operation to succeed.

        If SRP is supported and in operation, then the reserved
        bandwidth is determined by the operation of SRP, and any
        changes to the value of this object have no effect on the
        operational value of idleSlope(N).

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "12.20.1, 34.3"
    DEFVAL { 0 }
    ::= { ieee8021FqtssBapEntry 5 }

ieee8021FqtssAdminIdleSlopeLs OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "bits per second"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
```

```
            "The least significant 32 bits of the bandwidth,
            in bits per second, that the manager desires to allocate
            to the traffic class as  idleSlope(N). This object MUST be
            read or written at the same time as
            ieee8021FqtssAdminIdleSlopeMs,
            which represents the LS 32 bits of the value, in order
            for the read or write operation to succeed.

            If SRP is supported and in operation, then the reserved
            bandwidth is determined by the operation of SRP, and any
            changes to the value of this object have no effect on the
            operational value of idleSlope(N).

            The value of this object MUST be retained across
            reinitializations of the management system."
        REFERENCE   "12.20.1, 34.3"
        DEFVAL { 0 }
        ::= { ieee8021FqtssBapEntry 6 }

    ieee8021FqtssBapRowStatus OBJECT-TYPE
        SYNTAX RowStatus
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "Indicates the status of an entry (row) in this table, and is
            used to create/delete entries.

            The corresponding instances of the following objects
            must be set before this object can be made active(1):
               ieee8021FqtssBAPTrafficClass
               ieee8021FqtssDeltaBandwidth
               ieee8021FqtssOperIdleSlopeMs
               ieee8021FqtssOperIdleSlopeLs
               ieee8021FqtssAdminIdleSlopeMs
               ieee8021FqtssAdminIdleSlopeLs

            The corresponding instances of the following objects
            may not be changed while this object is active(1):
               ieee8021FqtssBAPTrafficClass"
    ::= { ieee8021FqtssBapEntry 7 }

    -- ============================================================
    -- The ieee8021FqtssMappings subtree
    -- This subtree defines the objects necessary for the assignment
    -- of transmission selection algorithms to traffic classes,
    -- and definition of regeneration table override values.
    -- ============================================================
```

```
-- ============================================================
-- the ieee8021FqtssTxSelectionAlgorithmTable
-- ============================================================

ieee8021FqtssTxSelectionAlgorithmTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssTxSelectionAlgorithmEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing the assignment of transmission
        selection algorithms to traffic classes for the Port.
        This table provides management of the Transmission
        Selection Algorithm Table defined in 8.6.8.

        For a given Port, a row in the table exists for each
        traffic class that is supported by the Port.

        The default assignments of transmission selection
        algorithms to traffic classes in the table are made
        on instantiation of the table, in accordance
        with the defaults defined in 8.6.8 and 34.5.

        All writable objects in this table must be
        persistent over power up restart/reboot."
    REFERENCE   "8.6.8, 12.20.2, 34.5"
    ::= { ieee8021FqtssMappings 1 }

ieee8021FqtssTxSelectionAlgorithmEntry OBJECT-TYPE
    SYNTAX      Ieee8021FqtssTxSelectionAlgorithmEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects that contain the mapping of a
        traffic class value to a transmission selection algorithm
        value."
    INDEX  { ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort,
             ieee8021FqtssTrafficClass  }
    ::= { ieee8021FqtssTxSelectionAlgorithmTable 1 }

Ieee8021FqtssTxSelectionAlgorithmEntry ::=
    SEQUENCE {
        ieee8021FqtssTrafficClass
            IEEE8021FqtssTrafficClassValue,
        ieee8021FqtssTxSelectionAlgorithmID
            IEEE8021FqtssTxSelectionAlgorithmIDValue
        }
```

```
ieee8021FqtssTrafficClass OBJECT-TYPE
    SYNTAX      IEEE8021FqtssTrafficClassValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The traffic class to which the transmission selection
         algorithm is assigned.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8, 12.20.2, 34.5"
    ::= { ieee8021FqtssTxSelectionAlgorithmEntry 1 }

ieee8021FqtssTxSelectionAlgorithmID OBJECT-TYPE
    SYNTAX      IEEE8021FqtssTxSelectionAlgorithmIDValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The identifier of the transmission selection algorithm
        assigned to the traffic class.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8, 12.20.2, 34.5"
    ::= { ieee8021FqtssTxSelectionAlgorithmEntry 2 }

-- ============================================================
-- the ieee8021FqtssSrpRegenOverrideTable
-- ============================================================

ieee8021FqtssSrpRegenOverrideTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssSrpRegenOverrideEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing the set of priority regeneration
        table override values for the Port.

        The recommended default values of priorities
        associated with SR classes, and the corresponding
        override values, are defined in 6.9.4.

        All writable objects in this table must be
        persistent over power up restart/reboot."
    REFERENCE   "35.1.4, 6.9.4, 12.20.3"
    ::= { ieee8021FqtssMappings 2 }

ieee8021FqtssSrpRegenOverrideEntry OBJECT-TYPE
```

```
    SYNTAX      Ieee8021FqtssSrpRegenOverrideEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects that contain the mapping of a
        priority value to a priority regeneration override
        value, and a boundary port indication.
        Rows in the table exist for all priorities that are
        associated with SR classes."
    INDEX  { ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort,
             ieee8021FqtssSrClassPriority  }
    ::= { ieee8021FqtssSrpRegenOverrideTable 1 }

Ieee8021FqtssSrpRegenOverrideEntry ::=
    SEQUENCE {
        ieee8021FqtssSrClassPriority
            IEEE8021PriorityValue,
        ieee8021FqtssPriorityRegenOverride
            IEEE8021PriorityValue,
        ieee8021FqtssSrpBoundaryPort
            TruthValue
        }

ieee8021FqtssSrClassPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The priority value that is overridden at the
        SRP domain boundary. "
    REFERENCE   "35.1.4, 6.9.4, 12.20.3"
    ::= { ieee8021FqtssSrpRegenOverrideEntry 1 }

ieee8021FqtssPriorityRegenOverride OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The priority value that is used to override the
         priority regeneration table entry at the SRP
         domain boundary.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "35.1.4, 6.9.4, 12.20.3"
    ::= { ieee8021FqtssSrpRegenOverrideEntry 2 }
```

```
ieee8021FqtssSrpBoundaryPort OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of the SRPdomainBoundaryPort parameter
        (35.1.4) for the priority. "
    REFERENCE   "35.1.4, 6.9.4, 12.20.3"
    ::= { ieee8021FqtssSrpRegenOverrideEntry 3 }

    -- ============================================================
    -- the ieee8021FqtssSRClassToPriorityTable
    -- ============================================================

    ieee8021FqtssSRClassToPriorityTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021FqtssSRClassToPriorityEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table containing the mapping of the SR Class to
            the associated priority.

            The default values for the entries of this table are
            specified in 34.5"
        REFERENCE   "12.40.4, 35.2.2.9.2, 6.9.3"
        ::= { ieee8021FqtssMappings 3 }

    ieee8021FqtssSRClassToPriorityEntry OBJECT-TYPE
        SYNTAX      Ieee8021FqtssSRClassToPriorityEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This attribute holds the Data Frame Priority
            (35.2.2.8.5(a)) value that will be used for streams
            that belong to the associated SR class."
        INDEX  { ieee8021BridgeBaseComponentId,
                 ieee8021BridgeBasePort,
                 ieee8021FqtssSrClassPriority }
        ::= { ieee8021FqtssSRClassToPriorityTable 1 }

    Ieee8021FqtssSRClassToPriorityEntry ::=
        SEQUENCE {
            ieee8021FqtssSRClassToPrioritySrClassID
                IEEE8021FqtssTrafficClassValue,
            ieee8021FqtssSRClassToPriorityRowStatus
                RowStatus
            }
```

```
            ieee8021FqtssSRClassToPrioritySrClassID OBJECT-TYPE
                SYNTAX      IEEE8021FqtssTrafficClassValue
                MAX-ACCESS  read-create
                STATUS      current
                DESCRIPTION
                    "The srClassId attribute provides the SR class ID
                    from Table 35-7 of 35.2.2.9.2, so that management
                    software can associate the traffic class to the
                    corresponding SR class A or B used by protocols
                    such as SRP.

                    The default values for this attribute use the
                    default values specified in 34.5 (i.e. Priority 3
                    for SRclassID 6 and Priority 2 for SRclassID 5).

                    If this managed object is not supported, the default
                    values specified in 34.5 are used as the fixed
                    configuration."
                REFERENCE   "12.20.4, 35.2.2.9.2 "
                ::= { ieee8021FqtssSRClassToPriorityEntry 1 }

            ieee8021FqtssSRClassToPriorityRowStatus OBJECT-TYPE
                SYNTAX RowStatus
                MAX-ACCESS read-create
                STATUS current
                DESCRIPTION
                  "Indicates the status of an entry (row) in this table,
and is
                    used to create/delete entries."
            ::= { ieee8021FqtssSRClassToPriorityEntry 2 }

ieee8021FqtssBapXTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssBapXEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of bandwidth availability
        parameters for each traffic class configured for use with time-
sensitive streams.
        All writable objects in this table must be
        persistent over power up restart/reboot."
    REFERENCE   "12.20.1"
    ::= { ieee8021FqtssBapX 1 }

ieee8021FqtssBapXEntry OBJECT-TYPE
    SYNTAX      Ieee8021FqtssBapXEntry
    MAX-ACCESS  not-accessible
    STATUS      current
```

```
    DESCRIPTION
        "A list of objects containing bandwidth allocation
        information for each traffic class configured for use with time-
sensitive streams. Rows in the table are
        automatically created and deleted as a result of the
        operation of the algorithm described in 34.5."
    AUGMENTS   { ieee8021FqtssBapEntry }
    ::= { ieee8021FqtssBapXTable 1 }

Ieee8021FqtssBapXEntry ::=
    SEQUENCE {
         ieee8021FqtssBAPClassMeasurementInterval
            Unsigned32,
         ieee8021FqtssBAPLockClassBandwidth
         TruthValue
    }

ieee8021FqtssBAPClassMeasurementInterval OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The value of the ClassMeasurementInterval
        parameter for the traffic class.
        This attribute uses units of nanoseconds,
        converted to/from units of seconds for use in
        34.3.

        If management of classMeasurementInterval is
        not supported, the default values (34.5) are
        used as the fixed Port configuration.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "12.20.1, 34.3, 34.4, 34.6"
    ::= { ieee8021FqtssBapXEntry 1 }

ieee8021FqtssBAPLockClassBandwidth OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This attribute determines the interpretation of
        deltaBandwidth. For the value false(2), deltaBandwidth
        is specified in 34.3.1. For true(1), deltaBandwidth is
        specified in 34.3.2"
    REFERENCE   "12.20.1, 34.3"
    DEFVAL { false }
```

```
    ::= { ieee8021FqtssBapXEntry 2 }

-- ================================================================
-- IEEE8021 FQTSS MIB - Conformance Information
-- ================================================================

ieee8021FqtssCompliances
    OBJECT IDENTIFIER ::= { ieee8021FqtssConformance 1 }
ieee8021FqtssGroups
    OBJECT IDENTIFIER ::= { ieee8021FqtssConformance 2 }

-- ================================================================
-- units of conformance
-- ================================================================

-- ================================================================
-- the ieee8021FqtssBap group
-- ================================================================

ieee8021FqtssBapGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssDeltaBandwidth,
        ieee8021FqtssOperIdleSlopeMs,
        ieee8021FqtssOperIdleSlopeLs,
        ieee8021FqtssAdminIdleSlopeMs,
        ieee8021FqtssAdminIdleSlopeLs,
        ieee8021FqtssBapRowStatus
    }
    STATUS      current
    DESCRIPTION
        "Objects that define bandwidth allocation for FQTSS."
    ::= { ieee8021FqtssGroups 1 }

-- ================================================================
-- the ieee8021FqtssTxSelectionAlgorithm group
-- ================================================================

ieee8021FqtssTxSelectionAlgorithmGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssTxSelectionAlgorithmID
    }
    STATUS      current
    DESCRIPTION
        "Objects that define transmission selection
        mappings for FQTSS."
    ::= { ieee8021FqtssGroups 2 }

-- ================================================================
```

```
-- the ieee8021FqtssBoundaryPort group
-- ============================================================

ieee8021FqtssBoundaryPortGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssPriorityRegenOverride,
        ieee8021FqtssSrpBoundaryPort
    }
    STATUS      current
    DESCRIPTION
        "Objects that define boundary port priority override
        mappings for FQTSS."
    ::= { ieee8021FqtssGroups 3 }

-- ============================================================
-- the ieee8021FqtssBapMeasurement group
-- ============================================================

ieee8021FqtssBapMeasurementGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssBAPClassMeasurementInterval,
        ieee8021FqtssBAPLockClassBandwidth
    }
    STATUS      current
    DESCRIPTION
        "Objects that define the SRP TSpec measurement interval
        and deltaBandwidth interpretation for FQTSS."
    ::= { ieee8021FqtssGroups 4 }

-- ============================================================
-- the ieee8021FqtssSRClassPriority group
-- ============================================================

ieee8021FqtssSRClassPriorityGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssSRClassToPrioritySrClassID,
        ieee8021FqtssSRClassToPriorityRowStatus
    }
    STATUS      current
    DESCRIPTION
        "Objects that define mappings of the SR class ID to
        the associated priority for FQTSS."
    ::= { ieee8021FqtssGroups 5 }

-- ============================================================
-- compliance statements
-- ============================================================
```

```
ieee8021FqtssCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting
        forwarding and queuing for time sensitive streams.

        Support of the objects defined in the IEEE8021-FQTSS MIB
        also requires support of the IEEE8021-BRIDGE-MIB; the
        provisions of 17.3.2 apply to implementations claiming
        support of the IEEE8021-FQTSS MIB. "

    MODULE -- this module
        MANDATORY-GROUPS {
            ieee8021FqtssBapGroup,
            ieee8021FqtssTxSelectionAlgorithmGroup,
            ieee8021FqtssBoundaryPortGroup
        }

    GROUP   ieee8021FqtssBapMeasurementGroup
    DESCRIPTION
        "Implementation of this group is optional. Implementation
        will allow management of the TSpec measurement interval
        and deltaBandwidth interpretation."

    GROUP   ieee8021FqtssSRClassPriorityGroup
    DESCRIPTION
        "Implementation of this group is optional. Implementation
        will allow management of the mapping of SR class IDs to the
        associated Priority."

    ::= { ieee8021FqtssCompliances 1 }

END
```

17.7.14 Definitions for the IEEE8021-SRP-MIB module

Replace 17.7.14 with the following text:

```
IEEE8021-SRP-MIB DEFINITIONS ::= BEGIN

-- ============================================================
-- MIB for support of IEEE 802.1Qat Stream Reservation Protocol
-- (SRP) in IEEE 802.1Q Bridges.
-- ============================================================

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Counter64,
    Unsigned32
        FROM SNMPv2-SMI
    MacAddress,
    TEXTUAL-CONVENTION,
    TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    ieee802dot1mibs,
    IEEE8021PriorityCodePoint,
    IEEE8021VlanIndex
        FROM IEEE8021-TC-MIB
    IEEE8021FqtssTrafficClassValue
        FROM IEEE8021-FQTSS-MIB
    ieee8021BridgeBaseComponentId,
    ieee8021BridgeBaseEntry,
    ieee8021BridgeBasePort,
    ieee8021BridgeBasePortEntry
        FROM IEEE8021-BRIDGE-MIB

    ;

ieee8021SrpMib MODULE-IDENTITY
    LAST-UPDATED "201810040000Z" -- October 4, 2018
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "WG-URL: http://ieee802.org/1/
         WG-EMail: STDS-802-1-L@IEEE.ORG

           Contact: IEEE 802.1 Working Group Chair
            Postal: C/O IEEE 802.1 Working Group
                    IEEE Standards Association
                    445 Hoes Lane
```

```
                Piscataway
                NJ 08854
                USA
        E-mail: STDS-802-1-L@IEEE.ORG"
DESCRIPTION
    "The Bridge MIB module for managing devices that support
    the IEEE Std 802.1Q Stream Reservation Protocol.

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

    Copyright (C) IEEE (2018).
    This version of this MIB module is part of IEEE Std 802.1Q;
    see the draft itself for full legal notices."

REVISION "201810040000Z" -- October 4, 2018
DESCRIPTION
        "Published as part of IEEE 802.1Qcc-2018.
              Added managed objects for Stream Reservation
              Protocol (SRP) Enhancements and Performance
              Improvements"

REVISION "201806280000Z" -- June 28, 2018
DESCRIPTION
        "Published as part of IEEE Std 802.1Q 2017.
        Cross references updated. "

REVISION "201512020000Z" -- December 2, 2015
DESCRIPTION
        "Published as part of IEEE Std 802.1Q-2014 Cor-1.
        ieee8021SrpReservationFailureBridgeId changed to
        ieee8021SrpReservationFailureSystemId."

REVISION "201412150000Z" -- December 15, 2014
DESCRIPTION
        "Published as part of IEEE Std 802.1Q 2014 revision.
        Cross references updated and corrected."

REVISION     "201102270000Z" -- February 27, 2011
DESCRIPTION
     "Minor edits to contact information etc. as part of
      2011 revision of Std 802.1Q."

REVISION     "201004190000Z" -- April 19, 2010
DESCRIPTION
     "Initial revision, included in IEEE 802.1Qat"
::= { ieee802dot1mibs 19 }
```

```
-- ============================================================
-- Textual Conventions
-- ============================================================

IEEE8021SrpStreamRankValue ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "An 802.1 SRP Stream Rank value. This is an integer,
         with the following interpretation placed on the value:

         0: Emergency, high-rank stream,
         1: Non-emergency stream."
    REFERENCE    "35.2.2.8.5b"
    SYNTAX       INTEGER {
                     emergency(0),
                     nonEmergency(1)
                 }


IEEE8021SrpStreamIdValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1x:1x:1x:1x:1x:1x.1x:1x"
    STATUS       current
    DESCRIPTION
            "Represents an SRP Stream ID, which is often defined
            as a MAC Address followed by a unique 16-bit ID."
    SYNTAX       OCTET STRING (SIZE (8))

IEEE8021SrpReservationDirectionValue ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "An 802.1 SRP Stream Reservation Direction value. This is
         an integer, with the following interpretation placed on
         the value:

         0: Talker registrations,
         1: Listener registrations."
    REFERENCE    "35.2.1.2"
    SYNTAX       INTEGER {
                     talkerRegistrations(0),
                     listenerRegistrations(1)
                 }


IEEE8021SrpReservationDeclarationTypeValue ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "An 802.1 SRP Stream Reservation Declaration Type value.
         This is an integer, with the following interpretation
```

```
        placed on the value:

        0: Talker Advertise,
        1: Talker Failed,
        2: Listener Asking Failed,
        3: Listener Ready,
        4: Listener Ready Failed."
    REFERENCE    "35.2.1.3"
    SYNTAX       INTEGER {
                     talkerAdvertise(0),
                     talkerFailed(1),
                     listenerAskingFailed(2),
                     listenerReady(3),
                     listenerReadyFailed(4)
                 }


IEEE8021SrpReservationFailureCodeValue ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "An 802.1 SRP Stream Reservation Failure Code value.
         This is an integer, with the following interpretation
         placed on the value:

         0: No failure,
         1: Insufficient bandwidth,
         2: Insufficient Bridge resources,
         3: Insufficient bandwidth for Traffic Class,
         4: StreamID in use by another Talker,
         5: Stream destination address already in use,
         6: Stream pre-empted by higher rank,
         7: Reported latency has changed,
         8: Egress port is not AVBCapable,
         9: Use a different destination_address,
         10: Out of MSRP resources,
         11: Out of MMRP resources,
         12: Cannot store destination_address,
         13: Requested priority is not an SR Class priority,
         14: MaxFrameSize is too large for media,
         15: maxFanInPorts limit has been reached,
         16: Changes in FirstValue for a registered StreamID,
         17: VLAN is blocked on this egress port (Registration
Forbidden),
        18: VLAN tagging is disabled on this egress port (untagged set),
         19: SR class priority mismatch."
    REFERENCE    "35.2.2.8.7"
    SYNTAX       INTEGER {
                    noFailure(0),
```

```
                    insufficientBandwidth(1),
                    insufficientResources(2),
                    insufficientTrafficClassBandwidth(3),
                    streamIDInUse(4),
                    streamDestinationAddressInUse(5),
                    streamPreemptedByHigherRank(6),
                    latencyHasChanged(7),
                    egressPortNotAVBCapable(8),
                    useDifferentDestinationAddress(9),
                    outOfMSRPResources(10),
                    outOfMMRPResources(11),
                    cannotStoreDestinationAddress(12),
                    priorityIsNoAnSRCLass(13),
                    maxFrameSizeTooLarge(14),
                    maxFanInPortsLimitReached(15),
                    firstValueChangedForStreamID(16),
                    vlanBlockedOnEgress(17),
                    vlanTaggingDisabledOnEgress(18),
                    srClassPriorityMismatch(19)
                }


-- ================================================================
-- subtrees in the SRP MIB
-- ================================================================

ieee8021SrpNotifications
    OBJECT IDENTIFIER ::= { ieee8021SrpMib 0 }

ieee8021SrpObjects
    OBJECT IDENTIFIER ::= { ieee8021SrpMib 1 }

ieee8021SrpConformance
    OBJECT IDENTIFIER ::= { ieee8021SrpMib 2 }

ieee8021SrpConfiguration
    OBJECT IDENTIFIER ::= { ieee8021SrpObjects 1 }

ieee8021SrpLatency
    OBJECT IDENTIFIER ::= { ieee8021SrpObjects 2 }

ieee8021SrpStreams
    OBJECT IDENTIFIER ::= { ieee8021SrpObjects 3 }

ieee8021SrpReservations
    OBJECT IDENTIFIER ::= { ieee8021SrpObjects 4 }
```

```
-- ============================================================
-- The ieee8021SrpConfiguration subtree
-- This subtree defines the objects necessary for the
-- operational management of SRP.
-- ============================================================

ieee8021SrpBridgeBaseTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpBridgeBaseEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table for SRP main control and status information.
        All writeable objects in this table must be persistent
        over power up restart/reboot.  These objects augment
        the ieee8021BridgeBasePortTable."
    ::= { ieee8021SrpConfiguration 1 }

ieee8021SrpBridgeBaseEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpBridgeBaseEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "SRP control and status information for a Bridge."
    AUGMENTS { ieee8021BridgeBaseEntry }
    ::= { ieee8021SrpBridgeBaseTable 1 }

Ieee8021SrpBridgeBaseEntry ::=
    SEQUENCE {
         ieee8021SrpBridgeBaseMsrpEnabledStatus
            TruthValue,
         ieee8021SrpBridgeBaseMsrpTalkerPruning
            TruthValue,
         ieee8021SrpBridgeBaseMsrpMaxFanInPorts
            Unsigned32,
         ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize
            Unsigned32,
         ieee8021SrpBridgeBaseMsrpTalkerVlanPruning
            TruthValue,
         ieee8021SrpBridgeBaseMsrpMaxSRClasses
            Unsigned32
    }

ieee8021SrpBridgeBaseMsrpEnabledStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The administrative status requested by management for
```

```
        MSRP.  The value true(1) indicates that MSRP should
        be enabled on this device, in all VLANs, on all ports
        for which it has not been specifically disabled.  When
        false(2), MSRP is disabled, in all VLANs and on all
        ports, and all MSRP frames will be forwarded
        transparently.  This objects affects both Applicant and
        Registrar state machines.  A transition from false(2)
        to true(1) will cause a reset of all MSRP state
        machines on all ports.

        This object may be modified while the corresponding
        instance of ieee8021BridgeBaseRowStatus is active(1).

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "35.2.1.4d"
    DEFVAL      { true }
    ::= { ieee8021SrpBridgeBaseEntry 1 }

ieee8021SrpBridgeBaseMsrpTalkerPruning OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The value of the talkerPruning parameter which
        controls the propagation of Talker declarations.
        The value true(1) indicates that Talker attributes
        are only declared on ports that have the Stream
        destination_address registered in the MMRP MAC
        Address Registration Entries.  When false(2),
        Talker attribute are declared on all egress ports
        in the active topology.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "12.22.1, 35.2.1.4b, 35.2.4.3.1"
    DEFVAL      { false }
    ::= { ieee8021SrpBridgeBaseEntry 2 }

ieee8021SrpBridgeBaseMsrpMaxFanInPorts OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The value of the msrpMaxFanInPorts parameter which
        limits the total number of ports on a Bridge that
        are allowed to establish reservations for inbound
        Streams.  A value of zero (0) indicates no fan-in
```

```
        limit is being specified and calculations involving
        fan-in will only be limited by the number of MSRP
        enabled ports.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "12.22.1, 35.2.1.4f"
    DEFVAL      { 0 }
    ::= { ieee8021SrpBridgeBaseEntry 3 }

ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The value of msrpLatencyMaxFrameSize parameter
        which is used in the calculation of the maximum
        latency through a Bridge.  The maximum size is
        defined to be 2000 octets by default, but may be
        set to a smaller or larger value dependent on the
        particular Bridge configuration. This parameter
        does not imply any type of policing of frame size,
        it is only used in the latency calculations.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "12.22.1, 35.2.1.4g"
    DEFVAL      { 2000 }
    ::= { ieee8021SrpBridgeBaseEntry 4 }

ieee8021SrpBridgeBaseMsrpTalkerVlanPruning OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This parameter allows to limit the Talker declaration
        to ports, that have the Stream's VLAN identifier
        registered as a member in the VLAN Registration
        Entries. The value true(1) indicates that Talker
        declarations are only sent out on ports, that have the
        Stream's VLAN identifier registered as a member in the
        VLAN Registration Entries. When false(2), Talker
        declarations are propagated according to the VLAN
        spanning tree."
    REFERENCE   "12.22.1, 35.2.1.4l"
    DEFVAL      { false }
    ::= { ieee8021SrpBridgeBaseEntry 5 }
```

```
ieee8021SrpBridgeBaseMsrpMaxSRClasses OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute provides the maximum number of SR classes
        supported by the Bridge."
    REFERENCE   "12.22.1, 35.2.1.4m"
    ::= { ieee8021SrpBridgeBaseEntry 6 }


ieee8021SrpBridgePortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpBridgePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table for SRP control and status information about
        every Bridge Port.  Augments the ieee8021BridgeBasePortTable."
    ::= { ieee8021SrpConfiguration 2 }

ieee8021SrpBridgePortEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpBridgePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "SRP control and status information for a Bridge Port."
    AUGMENTS { ieee8021BridgeBasePortEntry }
    ::= { ieee8021SrpBridgePortTable 1 }

Ieee8021SrpBridgePortEntry ::=
    SEQUENCE {
        ieee8021SrpBridgePortMsrpEnabledStatus
            TruthValue,
        ieee8021SrpBridgePortMsrpFailedRegistrations
            Counter64,
        ieee8021SrpBridgePortMsrpLastPduOrigin
            MacAddress,
        ieee8021SrpBridgePortSrPvid
            IEEE8021VlanIndex,
        ieee8021SrpBridgePortMsrpTalkerPrunningPerPort
        TruthValue
    }

ieee8021SrpBridgePortMsrpEnabledStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
```

```
        "The administrative state of MSRP operation on this port.  The
        value true(1) indicates that MSRP is enabled on this port
        in all VLANs as long as ieee8021BridgeMsrpEnabledStatus is
        also true(1).  A value of false(2) indicates that MSRP is
        disabled on this port in all VLANs: any MSRP frames received
        will be silently discarded, and no MSRP registrations will be
        propagated from other ports.  Setting this to a value of
        true(1) will be stored by the agent but will only take
        effect on the MSRP protocol operation if
        ieee8021BridgeMsrpEnabledStatus
        also indicates the value true(1).  This object affects
        all MSRP Applicant and Registrar state machines on this
        port.  A transition from false(2) to true(1) will
        cause a reset of all MSRP state machines on this port.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "35.2.1.4e"
    DEFVAL      { true }
    ::= { ieee8021SrpBridgePortEntry 1 }

ieee8021SrpBridgePortMsrpFailedRegistrations OBJECT-TYPE
    SYNTAX      Counter64
    UNITS       "failed MSRP registrations"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of failed MSRP registrations, for any
        reason, in all VLANs, on this port.

        Discontinuities in the value of the counter can occur at
        re-initialization of the management system, and at other
        times as indicated by the value of ifCounterDiscontinuityTime
        object of the associated interface (if any)."
    REFERENCE   "10.7.12.1"
    ::= { ieee8021SrpBridgePortEntry 2 }

ieee8021SrpBridgePortMsrpLastPduOrigin OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Source MAC Address of the last MSRP message
        received on this port."
    REFERENCE   "10.7.12.2"
    ::= { ieee8021SrpBridgePortEntry 3 }

ieee8021SrpBridgePortSrPvid OBJECT-TYPE
```

```
    SYNTAX      IEEE8021VlanIndex
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The default VLAN ID that Streams are assigned to.
        Talkers learn this VID from the SRP Domain attribute
        and tag Streams accordingly.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "35.2.2.8.3b"
    DEFVAL      { 2 }
    ::= { ieee8021SrpBridgePortEntry 4}

ieee8021SrpBridgePortMsrpTalkerPrunningPerPort OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This parameter controls the forwarding behavior for
        Talker declarations on the port when the TalkerPruning
        parameter is disabled for the bridge. The value true(1)
        indicates, that Talker declarations are only forwarded
        on that port, if the destination_address of the Stream
        is found in the MAC Address Registration Entries for the
        port. When false(2), Talker declarations are forwarded
        on that port regardless of the destination address."
    REFERENCE   "12.22.2, 35.2.1.4k"
    DEFVAL      { false }
    ::= { ieee8021SrpBridgePortEntry 5 }

-- ================================================================
-- The ieee8021SrpLatency subtree
-- This subtree defines the objects necessary for retrieving
-- the latency of the various traffic classes on a port.
-- ================================================================

-- ================================================================
-- the ieee8021SrpLatencyTable
-- ================================================================
ieee8021SrpLatencyTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpLatencyEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of latency measurement
        parameters for each traffic class."
    REFERENCE   "35.2.2.8.6"
```

```
    ::= { ieee8021SrpLatency 1 }

ieee8021SrpLatencyEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpLatencyEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing latency information
        for each traffic class. Rows in the table are
        automatically created for ports that are not an
        SRP domain boundary port (i.e. SRPdomainBoundaryPort
        is FALSE). See 35.1.4, 8.8.2, 12.22.3."
    INDEX  { ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort,
             ieee8021SrpTrafficClass  }
    ::= { ieee8021SrpLatencyTable 1 }

Ieee8021SrpLatencyEntry ::=
    SEQUENCE {
        ieee8021SrpTrafficClass
            IEEE8021FqtssTrafficClassValue,
        ieee8021SrpPortTcLatency
            Unsigned32
        }

ieee8021SrpTrafficClass OBJECT-TYPE
    SYNTAX      IEEE8021FqtssTrafficClassValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The traffic class number associated with the
        row of the table.

        Rows in the table are automatically created for
        ports that are not an SRP domain boundary port
        (i.e. SRPdomainBoundaryPort is FALSE)."
    REFERENCE   "35.1.4, 8.8.2, 12.22.3"
    ::= { ieee8021SrpLatencyEntry 1 }


ieee8021SrpPortTcLatency OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "nano-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of the portTcMaxLatency parameter for the
        traffic class.  This value is expressed in
```

```
        nano-seconds."
    REFERENCE   "35.2.1.4, 35.2.2.8.6"
    ::= { ieee8021SrpLatencyEntry 2 }


-- ================================================================
-- The ieee8021SrpStreams subtree
-- This subtree defines the objects necessary for retrieving
-- the characteristics of the various Streams currently registered.
-- ================================================================

-- ================================================================
-- the ieee8021SrpStreamTable
-- ================================================================
ieee8021SrpStreamTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpStreamEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of characteristics
        for each registered Stream."
    REFERENCE   "35.2.2.8"
    ::= { ieee8021SrpStreams 1 }

ieee8021SrpStreamEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpStreamEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing characteristics
        for each registered Stream. Rows in the table are
        automatically created for Streams registered on any
        port of a Bridge."
    INDEX  { ieee8021SrpStreamId }
    ::= { ieee8021SrpStreamTable 1 }

Ieee8021SrpStreamEntry ::=
    SEQUENCE {
        ieee8021SrpStreamId
            IEEE8021SrpStreamIdValue,
        ieee8021SrpStreamDestinationAddress
            MacAddress,
        ieee8021SrpStreamVlanId
            IEEE8021VlanIndex,
        ieee8021SrpStreamTspecMaxFrameSize
            Unsigned32,
        ieee8021SrpStreamTspecMaxIntervalFrames
            Unsigned32,
```

```
        ieee8021SrpStreamDataFramePriority
            IEEE8021PriorityCodePoint,
        ieee8021SrpStreamRank
            IEEE8021SrpStreamRankValue
        }

ieee8021SrpStreamId OBJECT-TYPE
    SYNTAX      IEEE8021SrpStreamIdValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Stream ID associated with the row of the table.

        Rows in the table are automatically created when
        Streams are registered via MSRP."
    REFERENCE   "35.2.2.8.2"
    ::= { ieee8021SrpStreamEntry 1 }

ieee8021SrpStreamDestinationAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The MAC destination address for the Stream described
        by this reservation."
    REFERENCE   "35.2.2.8.3a"
    ::= { ieee8021SrpStreamEntry 2}

ieee8021SrpStreamVlanId OBJECT-TYPE
    SYNTAX      IEEE8021VlanIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The VLAN ID associated with the MSRP registration
        for this Stream."
    REFERENCE   "35.2.2.8.3b"
    ::= { ieee8021SrpStreamEntry 3}

ieee8021SrpStreamTspecMaxFrameSize OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum size frame that will be sent by
        a Talker for this Stream. This value is part
        of the Traffic Specification for the Stream."
    REFERENCE   "35.2.2.8.4a"
    ::= { ieee8021SrpStreamEntry 4}
```

```
ieee8021SrpStreamTspecMaxIntervalFrames OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum number of frame that will be sent
        during a class measurement interval (L.2).  This
        value is part of the Traffic Specification for
        the Stream."
    REFERENCE   "35.2.2.8.4b, L.2"
    ::= { ieee8021SrpStreamEntry 5}

ieee8021SrpStreamDataFramePriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityCodePoint
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Priority Code Point (PCP) value that the
        referenced Stream will be tagged with. This value
        is used to distinguish Class A and Class B traffic."
    REFERENCE   "35.2.2.8.5a"
    ::= { ieee8021SrpStreamEntry 6}

ieee8021SrpStreamRank OBJECT-TYPE
    SYNTAX      IEEE8021SrpStreamRankValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "SRP supports emergency and non-emergency.
        Emergency traffic will interrupt non-emergency
        traffic if there is insufficient bandwidth or
        resources available for the emergency traffic."
    REFERENCE   "35.2.2.8.5b"
    ::= { ieee8021SrpStreamEntry 7}


    -- ============================================================
    -- the ieee8021SrpStreamPreloadTable
    -- ============================================================
    ieee8021SrpStreamPreloadTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021SrpStreamPreloadEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table containing a set of parameters for each StreamID
            that is preloaded on the Bridge as it initializes."
        REFERENCE   "12.22.6"
```

```
    ::= { ieee8021SrpStreams 2 }

ieee8021SrpStreamPreloadEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpStreamPreloadEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing characteristics
        for each registered Stream. Rows in the table are
        automatically created for Streams registered on any
        port of a Bridge."
    INDEX  { ieee8021SrpStreamPreloadId }
    ::= { ieee8021SrpStreamPreloadTable 1 }

Ieee8021SrpStreamPreloadEntry ::=
    SEQUENCE {
        ieee8021SrpStreamPreloadId
            IEEE8021SrpStreamIdValue,
        ieee8021SrpStreamPreloadDestinationAddress
            MacAddress,
        ieee8021SrpStreamPreloadVlanId
            IEEE8021VlanIndex,
        ieee8021SrpStreamPreloadTspecMaxFrameSize
            Unsigned32,
        ieee8021SrpStreamPreloadTspecMaxIntervalFrames
            Unsigned32,
        ieee8021SrpStreamPreloadDataFramePriority
            IEEE8021PriorityCodePoint,
        ieee8021SrpStreamPreloadRank
            IEEE8021SrpStreamRankValue
        }

ieee8021SrpStreamPreloadId OBJECT-TYPE
    SYNTAX      IEEE8021SrpStreamIdValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The 64-bit StreamID is used to match Talker
        registrations with their corresponding Listener
        registrations(35.2.4)."
    REFERENCE   "12.22.6, 35.2.2.8.2"
    ::= { ieee8021SrpStreamPreloadEntry 1 }

ieee8021SrpStreamPreloadDestinationAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
```

```
        "The MAC destination address for the Stream described
        by this reservation."
    REFERENCE   "12.22.6, 35.2.2.8.3a"
    ::= { ieee8021SrpStreamPreloadEntry 2}

ieee8021SrpStreamPreloadVlanId OBJECT-TYPE
    SYNTAX      IEEE8021VlanIndex
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The VLAN ID associated with the MSRP registration
        for this Stream."
    REFERENCE   "12.22.6, 35.2.2.8.3b"
    ::= { ieee8021SrpStreamPreloadEntry 3}

ieee8021SrpStreamPreloadTspecMaxFrameSize OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The maximum size frame that will be sent by
        a Talker for this Stream. This value is part
        of the Traffic Specification for the Stream."
    REFERENCE   "12.22.6, 35.2.2.8.4a"
    ::= { ieee8021SrpStreamPreloadEntry 4}

ieee8021SrpStreamPreloadTspecMaxIntervalFrames OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The maximum number of frames that the Talker may
        transmit in one classMeasurementInterval (34.3).
        This value is part of the Traffic Specification
        for the Stream."
    REFERENCE   "12.22.6, 35.2.2.8.4b"
    ::= { ieee8021SrpStreamPreloadEntry 5}

ieee8021SrpStreamPreloadDataFramePriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityCodePoint
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The Priority Code Point (PCP) value that the
        referenced Stream will be tagged with. This value
        is used to distinguish Class A and Class B traffic."
    REFERENCE   "12.22.6, 35.2.2.8.5a"
    ::= { ieee8021SrpStreamPreloadEntry 6}
```

```
    ieee8021SrpStreamPreloadRank OBJECT-TYPE
        SYNTAX      IEEE8021SrpStreamRankValue
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "SRP supports emergency and non-emergency.
            Emergency traffic will interrupt non-emergency
            traffic if there is insufficient bandwidth or
            resources available for the emergency traffic."
        REFERENCE   "12.22.6, 35.2.2.8.5b"
        ::= { ieee8021SrpStreamPreloadEntry 7}


-- ================================================================
-- The ieee8021SrpReservations subtree
-- This subtree defines the objects necessary for retrieving
-- the Stream attribute registrations on each port of a Bridge.
-- ================================================================

-- ================================================================
-- the ieee8021SrpReservationsTable
-- ================================================================
ieee8021SrpReservationsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpReservationsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing Stream attribute
        registrations per port."
    REFERENCE   "35.2.4"
    ::= { ieee8021SrpReservations 1 }

ieee8021SrpReservationsEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpReservationsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing Stream attribute
        registrations per port. Rows in the table are
        automatically created for Streams registered on any
        port of a Bridge."
    INDEX  { ieee8021SrpReservationStreamId,
             ieee8021SrpReservationDirection,
             ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort }
    ::= { ieee8021SrpReservationsTable 1 }
```

```
Ieee8021SrpReservationsEntry ::=
    SEQUENCE {
        ieee8021SrpReservationStreamId
            IEEE8021SrpStreamIdValue,
        ieee8021SrpReservationDirection
            IEEE8021SrpReservationDirectionValue,
        ieee8021SrpReservationDeclarationType
            IEEE8021SrpReservationDeclarationTypeValue,
        ieee8021SrpReservationAccumulatedLatency
            Unsigned32,
        ieee8021SrpReservationFailureSystemId
            OCTET STRING,
        ieee8021SrpReservationFailureCode
            IEEE8021SrpReservationFailureCodeValue,
        ieee8021SrpReservationDroppedStreamFrames
            Counter64,
        ieee8021SrpReservationStreamAge
            Unsigned32
        }

ieee8021SrpReservationStreamId OBJECT-TYPE
    SYNTAX      IEEE8021SrpStreamIdValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Stream ID associated with the row of the table.

        Rows in the table are automatically created when
        Streams are registered via MSRP."
    REFERENCE   "35.2.2.8.2"
    ::= { ieee8021SrpReservationsEntry 1 }

ieee8021SrpReservationDirection OBJECT-TYPE
    SYNTAX      IEEE8021SrpReservationDirectionValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The source of this Stream registration, either
        Talker or Listener."
    REFERENCE   "35.2.1.2"
    ::= { ieee8021SrpReservationsEntry 2 }

ieee8021SrpReservationDeclarationType OBJECT-TYPE
    SYNTAX      IEEE8021SrpReservationDeclarationTypeValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The type of Talker or Listener registration."
```

```
    REFERENCE   "35.2.1.3"
    ::= { ieee8021SrpReservationsEntry 3 }

ieee8021SrpReservationAccumulatedLatency OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "nano-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Accumulated Latency associated with the current
        registration.

        For Talker registrations this represents the accumulated
        latency from the Talker to the ingress port of this
        Bridge.

        For Listener registrations this represents the accumulated
        latency to the ingress port of the neighbor Bridge or
        end stations. This include the latency of the media
        attached to this egress port."
    REFERENCE   "35.2.2.8.6"
    ::= { ieee8021SrpReservationsEntry 4 }

ieee8021SrpReservationFailureSystemId OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(8))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The first system that changes a Talker Advertise to a
        Talker Failed registration will report its System
        Identification in this field. That single System
        Identification is then propagated from system to system."
    REFERENCE   "35.2.2.8.7a"
    ::= { ieee8021SrpReservationsEntry 5 }

ieee8021SrpReservationFailureCode OBJECT-TYPE
    SYNTAX      IEEE8021SrpReservationFailureCodeValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The first Bridge that changes a Talker Advertise to a
        Talker Failed registration will report the Failure Code
        in this field. That single Failure Code is then propagated
        from Bridge to Bridge."
    REFERENCE   "35.2.2.8.7b"
    ::= { ieee8021SrpReservationsEntry 6 }

ieee8021SrpReservationDroppedStreamFrames OBJECT-TYPE
```

```
    SYNTAX      Counter64
    UNITS       "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A count of the number of data stream frames that have
        been dropped for whatever reason.  These are not MSRP
        frames, but the stream data frames that are carried by
        the MSRP Reservation.

        Discontinuities in the value of the counter can occur at
        re-initialization of the management system, and at other
        times as indicated by the value of ifCounterDiscontinuityTime
        object of the associated interface (if any)."
     REFERENCE   "35.2.5.1"
    ::= { ieee8021SrpReservationsEntry 7 }

ieee8021SrpReservationStreamAge OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of seconds since the reservation was established
        on this port."
    REFERENCE   "35.2.1.4c"
    ::= { ieee8021SrpReservationsEntry 8 }


-- ================================================================
-- the ieee8021SrpReservationsPreloadTable
-- ================================================================
ieee8021SrpReservationsPreloadTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpReservationsPreloadEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing Stream attribute
        registrations per port."
    REFERENCE   "12.22.7"
    ::= { ieee8021SrpReservations 2 }

ieee8021SrpReservationsPreloadEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpReservationsPreloadEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing Stream attribute
```

```
        registrations per port. Rows in the table are
        automatically created for Streams registered on any
        port of a Bridge."
    INDEX  { ieee8021SrpReservationsPreloadStreamId,
             ieee8021SrpReservationPreloadDirection,
             ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort }
    ::= { ieee8021SrpReservationsPreloadTable 1 }

Ieee8021SrpReservationsPreloadEntry ::=
    SEQUENCE {
        ieee8021SrpReservationsPreloadStreamId
            IEEE8021SrpStreamIdValue,
        ieee8021SrpReservationPreloadDirection
            IEEE8021SrpReservationDirectionValue,
        ieee8021SrpReservationPreloadAccumulatedLatency
            Unsigned32
        }

ieee8021SrpReservationsPreloadStreamId OBJECT-TYPE
    SYNTAX      IEEE8021SrpStreamIdValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The 64-bit StreamID is used to match Talker
        registrations with their corresponding Listener
        registrations(35.2.4)."
    REFERENCE   "12.22.7, 35.2.2.8.2"
    ::= { ieee8021SrpReservationsPreloadEntry 1 }

ieee8021SrpReservationPreloadDirection OBJECT-TYPE
    SYNTAX      IEEE8021SrpReservationDirectionValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The source of this Stream registration, either
        Talker or Listener"
    REFERENCE   "12.22.7, 35.2.1.1"
    ::= { ieee8021SrpReservationsPreloadEntry 2 }

ieee8021SrpReservationPreloadAccumulatedLatency OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "nano-seconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The Accumulated Latency associated with the current
        registration.
```

```
            For Talker registrations this represents the accumulated
            latency from the Talker to the ingress port of this
            Bridge.

            For Listener registrations this represents the accumulated
            latency to the ingress port of the neighbor Bridge or
            end stations. This include the latency of the media
            attached to this egress port."
        REFERENCE   "12.22.7, 35.2.2.8.6"
        ::= { ieee8021SrpReservationsPreloadEntry 3 }


-- ================================================================
-- IEEE8021 SRP MIB - Conformance Information
-- ================================================================

ieee8021SrpCompliances
    OBJECT IDENTIFIER ::= { ieee8021SrpConformance 1 }
ieee8021SrpGroups
    OBJECT IDENTIFIER ::= { ieee8021SrpConformance 2 }

-- ================================================================
-- units of conformance
-- ================================================================

-- ================================================================
-- the ieee8021SrpConfiguration group
-- ================================================================

ieee8021SrpConfigurationGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpBridgeBaseMsrpEnabledStatus,
        ieee8021SrpBridgeBaseMsrpTalkerPruning,
        ieee8021SrpBridgeBaseMsrpMaxFanInPorts,
        ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize,
        ieee8021SrpBridgeBaseMsrpTalkerVlanPruning,
        ieee8021SrpBridgeBaseMsrpMaxSRClasses,
        ieee8021SrpBridgePortMsrpEnabledStatus,
        ieee8021SrpBridgePortMsrpFailedRegistrations,
        ieee8021SrpBridgePortMsrpLastPduOrigin,
        ieee8021SrpBridgePortSrPvid,
        ieee8021SrpBridgePortMsrpTalkerPrunningPerPort
    }
    STATUS      current
    DESCRIPTION
        "Objects that define configuration of SRP."
    ::= { ieee8021SrpGroups 1 }
```

```
-- ============================================================
-- the ieee8021SrpLatency group
-- ============================================================

ieee8021SrpLatencyGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpPortTcLatency
    }
    STATUS      current
    DESCRIPTION
        "Objects that define latency for SRP."
    ::= { ieee8021SrpGroups 2 }

-- ============================================================
-- the ieee8021SrpStreams group
-- ============================================================

ieee8021SrpStreamsGroup OBJECT-GROUP
    OBJECTS {
        -- ieee8021SrpStreamId,
        ieee8021SrpStreamDestinationAddress,
        ieee8021SrpStreamVlanId,
        ieee8021SrpStreamTspecMaxFrameSize,
        ieee8021SrpStreamTspecMaxIntervalFrames,
        ieee8021SrpStreamDataFramePriority,
        ieee8021SrpStreamRank
    }
    STATUS      current
    DESCRIPTION
        "Objects that define Streams for SRP."
    ::= { ieee8021SrpGroups 3 }

-- ============================================================
-- the ieee8021SrpReservations group
-- ============================================================

ieee8021SrpReservationsGroup OBJECT-GROUP
    OBJECTS {
        -- ieee8021SrpReservationStreamId,
        -- ieee8021SrpReservationDirection,
        ieee8021SrpReservationDeclarationType,
        ieee8021SrpReservationAccumulatedLatency,
        ieee8021SrpReservationFailureSystemId,
        ieee8021SrpReservationFailureCode,
        ieee8021SrpReservationDroppedStreamFrames,
        ieee8021SrpReservationStreamAge
    }
```

```
    STATUS      current
    DESCRIPTION
        "Objects that define Stream Reservations for SRP."
    ::= { ieee8021SrpGroups 4 }

-- ================================================================
-- the ieee8021SrpConfigurationPruning group
-- ================================================================

ieee8021SrpConfigurationPruningGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpBridgeBaseMsrpTalkerVlanPruning,
        ieee8021SrpBridgePortMsrpTalkerPrunningPerPort
    }
    STATUS      current
    DESCRIPTION
        "Objects that allow configuration of pruning behavior
        for SRP."
    ::= { ieee8021SrpGroups 5 }

-- ================================================================
-- the ieee8021SrpMonitoringSRclasses group
-- ================================================================

ieee8021SrpMonitoringSRclassesGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpBridgeBaseMsrpMaxSRClasses
    }
    STATUS      current
    DESCRIPTION
        "Objects that provides information on the maximum number
        of SR classes supported on the Bridge."
    ::= { ieee8021SrpGroups 6 }

-- ================================================================
-- the ieee8021SrpStreamsPreload group
-- ================================================================

ieee8021SrpStreamsPreloadGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpStreamPreloadId,
        ieee8021SrpStreamPreloadDestinationAddress,
        ieee8021SrpStreamPreloadVlanId,
        ieee8021SrpStreamPreloadTspecMaxFrameSize,
        ieee8021SrpStreamPreloadTspecMaxIntervalFrames,
        ieee8021SrpStreamPreloadDataFramePriority,
        ieee8021SrpStreamPreloadRank
    }
```

```
    STATUS      current
    DESCRIPTION
        "Objects that allow to preload parameters for each
        StreamId on Bridge Ports as the Bridge initializes."
    ::= { ieee8021SrpGroups 7 }

-- ============================================================
-- the ieee8021SrpReservationsPreload group
-- ============================================================

ieee8021SrpReservationsPreloadGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpReservationsPreloadStreamId,
        ieee8021SrpReservationPreloadDirection,
        ieee8021SrpReservationPreloadAccumulatedLatency
    }
    STATUS      current
    DESCRIPTION
        "Objects that allow to initialize Streams within each
        Bridge as it powers up, to preload the Stream
        registrations that will later be provided by operation
        of SRP."
    ::= { ieee8021SrpGroups 8 }

-- ============================================================
-- compliance statements
-- ============================================================

ieee8021SrpCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting
        Stream Reservation Protocol.

        Support of the objects defined in the IEEE8021-SRP MIB
        also requires support of the IEEE8021-BRIDGE-MIB; the
        provisions of 17.3.2 apply to implementations claiming
        support of the IEEE8021-SRP MIB."

    MODULE -- this module
        MANDATORY-GROUPS {
            ieee8021SrpConfigurationGroup,
            ieee8021SrpLatencyGroup,
            ieee8021SrpStreamsGroup,
            ieee8021SrpReservationsGroup
        }

    GROUP   ieee8021SrpConfigurationPruningGroup
```

```
    DESCRIPTION
        "Implementation of this group is optional. Implementation
        will allow configuration of pruning behavior for SRP."

    GROUP   ieee8021SrpMonitoringSRclassesGroup
    DESCRIPTION
        "Implementation of this group is optional. Implementation
        will allow configuration of pruning behavior for SRP."

    GROUP   ieee8021SrpStreamsPreloadGroup
    DESCRIPTION
        "Implementation of this group is optional. Implementation
        will allow to preload parameters for each StreamId on
        Bridge Ports as the Bridge initializes."

    GROUP   ieee8021SrpReservationsPreloadGroup
    DESCRIPTION
        "Implementation of this group is optional. Implementation
        will allow to initialize Streams within each Bridge as it
        powers up, to preload the Stream registrations that will
        later be provided by operation of SRP."

    ::= { ieee8021SrpCompliances 1 }

END
```

Insert the following subclause (17.7.25) after 17.7.24:

17.7.25 Definitions for the IEEE8021-SR-RM-MIB module

```
IEEE8021-TSN-REMOTE-MANAGEMENT-MIB DEFINITIONS ::= BEGIN

-- ============================================================
-- MIB for support of 802.1Qcc Stream Reservation Protocol
-- (SRP) Enhancements and Performance Improvements in
-- 802.1Q Bridges.
-- ============================================================

IMPORTS
   OBJECT-GROUP,
   MODULE-COMPLIANCE
      FROM SNMPv2-CONF
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Counter64,
    Unsigned32
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION,
    TruthValue,
    RowStatus
        FROM SNMPv2-TC
    IEEE8021BridgePortNumber,
    ieee802dot1mibs
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBasePort,
    ieee8021BridgeBaseComponentId,
    ieee8021BridgeTrafficClass
        FROM IEEE8021-BRIDGE-MIB
    ieee8021QBridgeVlanIndex
        FROM IEEE8021-Q-BRIDGE-MIB
    ;

ieee8021TsnRemoteMgmtMib MODULE-IDENTITY
    LAST-UPDATED "201810040000Z" -- October 4, 2018
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "  WG-URL: http://ieee802.org/1/
         WG-EMail: STDS-802-1-L@IEEE.ORG

           Contact: IEEE 802.1 Working Group Chair
            Postal: C/O IEEE 802.1 Working Group
                    IEEE Standards Association
                    445 Hoes Lane
                    Piscataway
                    NJ 08854
                    USA
           E-mail: STDS-802-1-L@IEEE.ORG"
    DESCRIPTION
        "The Bridge MIB module for managing devices that support
        the IEEE Std 802.1Q Stream Reservation Protocol Enhancements
```

```
        and Performance Improvements.

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

        Copyright (C) IEEE (2018).
        This version of this MIB module is part of IEEE802.1Q;
        see the draft itself for full legal notices."

    REVISION "201810040000Z" -- October 4, 2018
    DESCRIPTION
        "Initial revision, included in IEEE 802.1Qcc-2018"
    ::= { ieee802dot1mibs 32 }

-- ================================================================
-- subtrees in the TSN Remote Management MIB
-- ================================================================

ieee8021TsnRemoteMgmtNotifications
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtMib 0 }

ieee8021TsnRemoteMgmtObjects
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtMib 1 }

ieee8021TsnRemoteMgmtConformance
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtMib 2 }

ieee8021TsnRemoteMgmtBridgeDelay
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtObjects 1 }

ieee8021TsnRemoteMgmtPropagationDelay
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtObjects 2 }

ieee8021TsnRemoteMgmtStaticTrees
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtObjects 3 }

ieee8021TsnRemoteMgmtMrpExternalControl
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtObjects 4 }

-- ================================================================
-- the ieee8021TsnRemoteBridgeDelayTable
-- ================================================================

ieee8021TsnRemoteMgmtBridgeDelayTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021TsnRemoteMgmtBridgeDelayEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of parameters necessary to
        determine the delay of frames as they pass through the
        Bridge's relay.
        There is one Bridge Delay managed object per Port pair of
        a Bridge component. The Port pair consists of three indices,
        an ingress Port followed by an egress Port and a traffic
```

```
            class associated with the Port pair."
        REFERENCE   "12.32.1"
        ::= { ieee8021TsnRemoteMgmtBridgeDelay 1 }

    ieee8021TsnRemoteMgmtBridgeDelayEntry OBJECT-TYPE
        SYNTAX      Ieee8021TsnRemoteMgmtBridgeDelayEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A list of objects containing information necessary to
            determine the delay of frames as they pass through the
            Bridge's relay."
        INDEX  { ieee8021BridgeBaseComponentId,
                 ieee8021BridgeTrafficClass,
                 ieee8021TsnRemoteMgmtBridgeIngressPort,
                 ieee8021TsnRemoteMgmtBridgeEgressPort
               }
        ::= { ieee8021TsnRemoteMgmtBridgeDelayTable 1 }

    Ieee8021TsnRemoteMgmtBridgeDelayEntry ::=
        SEQUENCE {
            ieee8021TsnRemoteMgmtBridgeIngressPort
                IEEE8021BridgePortNumber,
            ieee8021TsnRemoteMgmtBridgeEgressPort
                IEEE8021BridgePortNumber,
            ieee8021TsnRemoteMgmtIndependentDelayMin
                Unsigned32,
            ieee8021TsnRemoteMgmtIndependentDelayMax
                Unsigned32,
           ieee8021TsnRemoteMgmtDependentDelayMin
                Unsigned32,
            ieee8021TsnRemoteMgmtDependentDelayMax
                Unsigned32
        }

    ieee8021TsnRemoteMgmtBridgeIngressPort OBJECT-TYPE
        SYNTAX      IEEE8021BridgePortNumber
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "..."
        REFERENCE   "..."
        ::= { ieee8021TsnRemoteMgmtBridgeDelayEntry 1 }

    ieee8021TsnRemoteMgmtBridgeEgressPort OBJECT-TYPE
        SYNTAX      IEEE8021BridgePortNumber
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "..."
        REFERENCE   "..."
        ::= { ieee8021TsnRemoteMgmtBridgeDelayEntry 2 }

    ieee8021TsnRemoteMgmtIndependentDelayMin OBJECT-TYPE
```

```
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute provides the minimum delay independent
        from frame length for a frame to forward from ingress
        port to egress port.

        The delay begins when the message timestamp point of the
        ingress frame passes the reference plane marking the
        boundary between the network media and PHY. The delay
        ends when the message timestamp point of the egress
        frame passes the reference plane marking the boundary
        between the network media and PHY. The message timestamp
        point is specified by IEEE Std 802.1AS for various media,
        near the start of the frame.

        Note: This delay includes all aspects of length-independent
        delay for a frame that is forwarded, including handling of
        error conditions."
    REFERENCE   "12.32.1.1"
    ::= { ieee8021TsnRemoteMgmtBridgeDelayEntry 3 }

ieee8021TsnRemoteMgmtIndependentDelayMax OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute provides the maximum delay independent
        from frame length for a frame to forward from ingress
        port to egress port.

        The delay begins when the message timestamp point of
        the ingress frame passes the reference plane marking
        the boundary between the network media and PHY. The delay
        ends when the message timestamp point of the egress frame
        passes the reference plane marking the boundary between
        the network media and PHY. The message timestamp point is
        specified by IEEE Std 802.1AS for various media, near the
        start of the frame."
    REFERENCE   "12.32.1.1"
    ::= { ieee8021TsnRemoteMgmtBridgeDelayEntry 4 }

ieee8021TsnRemoteMgmtDependentDelayMin OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute provides the minimum length-dependent
        delay from ingress port to egress port.

        It provides the portion of delay that is dependent on
        frame length, where frame length is the number of octets
        that transfer across the MAC Service interfaces. Each
```

```
        length-dependent delay attribute specifies the time for
        a single octet of the frame to transfer from ingress
        to egress."
    REFERENCE   "12.32.1.2"
    ::= { ieee8021TsnRemoteMgmtBridgeDelayEntry 5 }

ieee8021TsnRemoteMgmtDependentDelayMax OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute provides the maximum length-dependent
        delay from ingress port to egress port.

        It provides the portion of delay that is dependent on
        frame length, where frame length is the number of octets
        that transfer across the MAC Service interfaces. Each
        length-dependent delay attribute specifies the time for
        a single octet of the frame to transfer from ingress
        to egress."
    REFERENCE   "12.32.1.2"
    ::= { ieee8021TsnRemoteMgmtBridgeDelayEntry 6 }

-- ============================================================
-- the ieee8021TsnRemoteMgmtPropagationDelayTable
-- ============================================================

ieee8021TsnRemoteMgmtPropagationDelayTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021TsnRemoteMgmtPropagationDelayEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of parameters necessary to
        determine the delay along the network media (e.g. cable)
        for a frame transmitted from the specifed Port of this
        Bridge to the neighboring Port on a different Bridge.
        There is one Propagation Delay managed object per
        egress Port of a Bridge."
    REFERENCE   "12.32.2"
    ::= { ieee8021TsnRemoteMgmtPropagationDelay 1 }

ieee8021TsnRemoteMgmtPropagationDelayEntry OBJECT-TYPE
    SYNTAX      Ieee8021TsnRemoteMgmtPropagationDelayEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing information necessary to
            determine the delay along the network media (e.g. cable)
            for a frame transmitted from the specifed Port of this
            Bridge to the neighboring Port on a different Bridge."
    INDEX  { ieee8021BridgeBasePort }
    ::= { ieee8021TsnRemoteMgmtPropagationDelayTable 1 }

Ieee8021TsnRemoteMgmtPropagationDelayEntry ::=
```

```
    SEQUENCE {
        ieee8021TsnRemoteMgmtTxPropagationDelay
            Unsigned32
        }

ieee8021TsnRemoteMgmtTxPropagationDelay OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute provides the transmission propagation delay.

        The propagation delay begins when the message timestamp
        point of an egress frame passes the reference plane marking
        the boundary between the network media and PHY. It ends
        when the message timestamp point of an ingress frame on the
        neighboring Bridge's Port passes the reference plane marking
        the boundary between the network media and PHY. The message
        timestamp point is specified by IEEE Std 802.1AS for
        various media."
    REFERENCE   "12.32.2.1"
    ::= { ieee8021TsnRemoteMgmtPropagationDelayEntry 1 }


-- ============================================================
-- The Static Tree subtree
-- This subtree defines the objects necessary to determine if
-- the static trees feature is supported by the Bridge.
-- ============================================================

ieee8021TsnRemoteMgmtStaticTreesSupported OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used by the TSN CNC to determine that
        TE-MSTID is supported by the Bridge."
    REFERENCE   "12.32.3.1"
    ::= { ieee8021TsnRemoteMgmtStaticTrees 1 }

-- ============================================================
-- the ieee8021TsnRemoteMgmtMsrpMrpExternalControlTable
-- ============================================================

ieee8021TsnRemoteMgmtMsrpMrpExternalControlTable OBJECT-TYPE
     SYNTAX       SEQUENCE OF Ieee8021TsnRemoteMgmtMsrpMrpExternalContro-
lEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of parameters necessary for
        a network manager to 1) disable MRP attribute propagation (MAP)
        for the MRP Participant of a bridge port, 2) read MRP attribute
        registrations that the MRP Participant receives, and 3) write
```

```
            MRP attribute values for the MRP Participant to declare."
        REFERENCE   "12.32.4"
        ::= { ieee8021TsnRemoteMgmtMrpExternalControl 1 }

    ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry OBJECT-TYPE
        SYNTAX      Ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A list of objects necessary for a network manager to
            1) disable MRP attribute propagation (MAP) for the
            MRP Participant of a bridge port, 2) read MRP attribute
            registrations that the MRP Participant receives, and 3) write
            MRP attribute values for the MRP Participant to declare."
        INDEX  { ieee8021BridgeBaseComponentId,
                 ieee8021BridgeBasePort,
                 ieee8021QBridgeVlanIndex }
        ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlTable 1 }

    Ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry ::=
        SEQUENCE {
            ieee8021TsnRemoteMgmtMsrpMrpExternalControl
                TruthValue,
            ieee8021TsnRemoteMgmtMrpIndicationList
                OCTET STRING,
            ieee8021TsnRemoteMgmtMrpIndicationListLength
                Unsigned32,
            ieee8021TsnRemoteMgmtMrpIndicationChangeCounter
                Counter64,
            ieee8021TsnRemoteMgmtMrpAdminRequestList
                OCTET STRING,
            ieee8021TsnRemoteMgmtMrpAdminRequestListLength
                Unsigned32,
            ieee8021TsnRemoteMgmtMrpOperRequestList
                OCTET STRING,
            ieee8021TsnRemoteMgmtMrpOperRequestListLength
                Unsigned32
            }

    ieee8021TsnRemoteMgmtMsrpMrpExternalControl OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "This attribute is used to indicate, whether MRP
            attributes are propagated on the MRP Participant,
            according to the specifications for MRP Attribute
            Propagation (MAP) and specifications of the
            MRP Application. When true(1), the MRP Participant is
            removed from the MRP Application's MAP Context. The
            MRP Participant performs all other aspects of MRP,
            including MRP operation, MRP specifications, and
            MRPDU encodings. The application component stores MAD
            indications for registration received on the Port,
```

```
        and invokes MAD requests for declarations on the Port.
        When false(2), MRP attributes propagate on the
        MRP Participant according to the specifications for
        MRP Attribute Propagation (MAP) and specifications of
        the MRP Application. Ports with the externalControl
        attribute false(2) are considered as candidates for
        the MRP Application's MAP Context. The remaining
        attributes of this subtree are ignored by Ports with
        the externalControl attribute false(2).
        This managed object applies to the MSRP application.
        A table is provided for each MAP Context (VLAN ID)."
    REFERENCE   "12.32.4.1"
    DEFVAL { false }
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 1 }

ieee8021TsnRemoteMgmtMrpIndicationList OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to store the list of all joined
        MRP attributes for the MRP Participant when the
        ieee8021TsnRemoteMgmtMrpExternalControl attribute is
        true(1). When the ieee8021TsnRemoteMgmtMrpExternalControl
        attribute is false(2), this attribute is ignored by the
        MRP Participant, and returns the empty octet string."
    REFERENCE   "12.32.4.2"
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 2 }

ieee8021TsnRemoteMgmtMrpIndicationListLength OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to provide the number of octets
        in the ieee8021TsnRemoteMgmtMrpIndicationListLength
        attribute. When the ieee8021TsnRemoteMgmtMrpExternalControl
        attribute is false(2), this attribute returns zero."
    REFERENCE   "12.32.4.3"
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 3 }

ieee8021TsnRemoteMgmtMrpIndicationChangeCounter OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to provide the number of changes
        done to the ieee8021TsnRemoteMgmtMrpIndicationList. When
        the ieee8021TsnRemoteMgmtMrpExternalControl attribute is
        false(2), this attribute returns zero."
    REFERENCE   "12.34.4.4"
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 4 }

ieee8021TsnRemoteMgmtMrpAdminRequestList OBJECT-TYPE
```

```
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to provide the administrative
        value of the current list of MAD requests for the
        MRP Participant (operRequestList). Each entry in this
        attribute is encoded as the attribute_type parameter as
        a single octet, followed by the length of the
        attribute_value parameter as a single octet, followed
        by a sequence of octets for the attribute_value parameter.
        When the ieee8021TsnRemoteMgmtMrpExternalControl attribute
        is true(1), this attribute is copied to the
        ieee8021TsnRemoteMgmtMrpOperRequestList attribute as soon
        as possible according to the implementation. When the
        ieee8021TsnRemoteMgmtMrpExternalControl attribute is
        false(2), this attribute is ignored by the MRP Participant,
        but it retains its value."
    REFERENCE   "12.32.4.5"
    DEFVAL { "" }
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 5 }

ieee8021TsnRemoteMgmtMrpAdminRequestListLength OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to provide the administrative
        value for the number of octets in the
        ieee8021TsnRemoteMgmtMrpAdminRequestList attribute.
        When the ieee8021TsnRemoteMgmtMrpExternalControl attribute
        is true(1), this attribute is copied to the
        ieee8021TsnRemoteMgmtMrpOperRequestListLength attribute at
        the same time that the ieee8021TsnRemoteMgmtMrpAdminRequestList
         attribute is copied to the ieee8021TsnRemoteMgmtMrpOperRequest-
List
        attribute. When the ieee8021TsnRemoteMgmtMrpExternalControl
        attribute is false(2), this attribute is ignored by the
        MRP Participant, but it retains its value."
    REFERENCE   "12.32.4.6"
    DEFVAL { 0 }
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 6 }

ieee8021TsnRemoteMgmtMrpOperRequestList OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to proivde the operational value
        of the current list of MAD requests for the MRP Participant."
    REFERENCE   "12.32.4.7"
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 7 }

ieee8021TsnRemoteMgmtMrpOperRequestListLength OBJECT-TYPE
```

```
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This attribute is used to provide the operational value
        of the ieee8021TsnRemoteMgmtMrpAdminRequestListLength
        attribute, and it is copied at the same time that
        ieee8021TsnRemoteMgmtMrpAdminRequestList attribute is
        copied to ieee8021TsnRemoteMgmtMrpOperRequestList."
    REFERENCE   "12.32.4.8"
    ::= { ieee8021TsnRemoteMgmtMsrpMrpExternalControlEntry 8 }


-- ================================================================
-- IEEE802 TSN REMOTE MANAGEMENT MIB - Conformance Information
-- ================================================================

ieee8021TsnRemoteMgmtCompliances
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtConformance 1 }
ieee8021TsnRemoteMgmtGroups
    OBJECT IDENTIFIER ::= { ieee8021TsnRemoteMgmtConformance 2 }

-- ================================================================
-- units of conformance
-- ================================================================

-- ================================================================
-- the ieee8021TsnRemoteMgmtBridgeDelay group
-- ================================================================

ieee8021TsnRemoteMgmtBridgeDelayGroup OBJECT-GROUP
    OBJECTS {
        ieee8021TsnRemoteMgmtIndependentDelayMin,
        ieee8021TsnRemoteMgmtIndependentDelayMax,
        ieee8021TsnRemoteMgmtDependentDelayMin,
        ieee8021TsnRemoteMgmtDependentDelayMax
    }
    STATUS      current
    DESCRIPTION
        "Objects that define the delay of frames as they pass
        through the Bridge's relay."
    ::= { ieee8021TsnRemoteMgmtGroups 1 }


-- ================================================================
-- the ieee8021TsnRemoteMgmtPropagationDelay group
-- ================================================================

ieee8021TsnRemoteMgmtPropagationDelayGroup OBJECT-GROUP
    OBJECTS {
        ieee8021TsnRemoteMgmtTxPropagationDelay
    }
    STATUS      current
    DESCRIPTION
```

```
        "Objects that define delay of frames along the network
        media (e.g. cable)."
    ::= { ieee8021TsnRemoteMgmtGroups 2 }


-- ================================================================
-- the ieee8021TsnRemoteMgmtStaticTrees group
-- ================================================================

ieee8021TsnRemoteMgmtStaticTreesGroup OBJECT-GROUP
    OBJECTS {
        ieee8021TsnRemoteMgmtStaticTreesSupported
    }
    STATUS      current
    DESCRIPTION
        "Objects that define static tree support."
    ::= { ieee8021TsnRemoteMgmtGroups 3 }


-- ================================================================
-- the ieee8021TsnRemoteMgmtMrpExternalControl group
-- ================================================================

ieee8021TsnRemoteMgmtMrpExternalControlGroup OBJECT-GROUP
    OBJECTS {
        ieee8021TsnRemoteMgmtMsrpMrpExternalControl,
        ieee8021TsnRemoteMgmtMrpIndicationList,
        ieee8021TsnRemoteMgmtMrpIndicationListLength,
        ieee8021TsnRemoteMgmtMrpIndicationChangeCounter,
        ieee8021TsnRemoteMgmtMrpAdminRequestList,
        ieee8021TsnRemoteMgmtMrpAdminRequestListLength,
        ieee8021TsnRemoteMgmtMrpOperRequestList,
        ieee8021TsnRemoteMgmtMrpOperRequestListLength
    }
    STATUS      current
    DESCRIPTION
        "Objects that define configuration of MRP External control."
    ::= { ieee8021TsnRemoteMgmtGroups 4 }


-- ================================================================
-- compliance statements
-- ================================================================

ieee8021TsnRemoteMgmtCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting
        TSN Remote management.

        Support of the objects defined in the IEEE8021-TSN REMOTE
        MANAGEMENT MIB also requires support of the IEEE8021-BRIDGE-MIB;
        the provisions of 17.3.2 apply to implementations claiming
        support of the IEEE8021-TSN REMOTE  MANAGEMENT MIB."
```

```
    MODULE -- this module
        MANDATORY-GROUPS {
            ieee8021TsnRemoteMgmtBridgeDelayGroup,
            ieee8021TsnRemoteMgmtPropagationDelayGroup,
            ieee8021TsnRemoteMgmtStaticTreesGroup,
            ieee8021TsnRemoteMgmtMrpExternalControlGroup
        }

    ::= { ieee8021TsnRemoteMgmtCompliances 1 }

END
```

34. Forwarding and queuing enhancements for time-sensitive streams (FQTSS)

Change the text of Clause 34 as shown:

34.1 Overview

This clause describes a set of tools that can be used to support the forwarding and queuing requirements of time-sensitive streams. In this context, a ~~"~~time-sensitive stream~~"~~ is taken to be a stream (flow) of traffic, transmitted from a single source station (Talker), destined for one or more destination stations (Listeners), where the traffic is sensitive to timely delivery, and in particular, requires transmission latency to be bounded. Such streams include video or audio data streams, where there is a desire to limit the amount of buffering required in the receiving station (Listener).

NOTE 1—An example of this requirement would be a live performance where a video image of the performance is simultaneously projected on a screen in the auditorium, and it is desirable for the sound and image to be "in sync" with the performance.

The term "stream reservation class" (SR class) refers to a traffic class whose bandwidth can be reserved for time-sensitive streams using the Stream Reservation Protocol (SRP). FQTSS specifies the relationship between an SR class and the underlying tools of this standard, such as traffic class, priority, and transmission selection.

NOTE 2—The FQTSS functions described in this clause are intended for use with SRP, with the exception of adminIdleSlope [item c) in 34.3]. The adminIdleSlope parameter is intended for TSN configuration models that do not use SRP. In the fully centralized model of TSN configuration (46.1.3.3), the CNC entity can use adminIdleSlope to reserve bandwidth for a traffic class.

In order to address the needs of such traffic in Bridges, the following are provided:

a) A means of detecting the boundary between a set of Bridges that support SRP (an SRP domain) and surrounding Bridges that do not support SRP. This mechanism is described in 34.2.

~~NOTE 2—The primary intent of these functions is to support SRP; however, there is no specific interdependency between these functions and SRP, so they could equally be used to support other admission control mechanisms if they were implemented.~~

b) A set of bandwidth availability parameters for each ~~p~~Port that are used to record the maximum bandwidth available to a given outbound queue, and the actual bandwidth reserved, for that queue. These parameters are described in 34.3.

~~c) A credit-based shaper algorithm, defined in 8.6.8.1, that is used to shape the transmission of stream-based traffic in accordance with the bandwidth that has been reserved on a given outbound queue.~~

c) ~~d)~~ A discussion of the relationship between the size of the layer 2 "payload" (the MSDU) carried in a frame and how that relates to the actual bandwidth that will be consumed when that MSDU is transmitted on a particular Port (34.4).

d) ~~e) An algorithm for determining~~ Specifications for the default configuration of SR classes, including the mapping of the priorities associated with received frames onto the traffic classes available on the transmission Ports of a Bridge (34.5).

e) ~~f) A definition of the required behavior of an end station that acts as the source of a time-sensitive data stream~~ Specifications for use of transmission selection algorithms for SR classes (34.6).

34.2 Detection of SRP domains

The concepts of ~~audio/video (AV)~~ time-sensitive streams, the Stream Reservation Protocol (SRP), and the traffic forwarding and shaping functions that support stream transmission (~~see 6.9.4 and~~e.g., 8.6.8.1) rely on the ability of each Bridge to detect whether each of its ~~p~~Ports is at the edge of a region of connected Bridges that support SRP on a particular priority, so that the Priority Code Point values associated with traffic entering an *SRP domain* (3.257) can be properly regenerated at the boundary of the domain, as described in 6.9.4.

Bridges detect the edge of an SRP domain by observing SRP behavior. If a Bridge receives SRP registrations using a particular priority, then it is reasonable to believe that they are being received from an SRP capable device; the SRP engine can therefore signal which Ports of a Bridge are at the boundary of an SRP domain. The per-~~p~~Port, per-SR class, *SRPdomainBoundaryPort* parameter determines whether a Port is considered to be at the edge of an SRP domain or within the core of the domain, as defined in 35.1.4. This parameter is controlled by the operation of SRP.

NOTE—SRP domain detection is based on the assumption that a device connected to a Port either is SRP capable for a given SR class, or is not SRP capable for that SR class. SRP provides a boundary detection mechanism through the exchange of MSRPDUs; the boundary of a domain therefore expands to include Ports as SRP attributes are declared. The position of the domain boundary has no effect on the transmission of SRP frames; rather, it reflects where SRP activity is occurring. Ports are removed from the SRP domain when they are removed from the active topology.

34.3 The bandwidth availability parameters

The following bandwidth availability parameters exist for each Port, and for each traffic class, N, that ~~supports the credit-based shaper algorithm~~is configured as an SR class (34.5):

a) ***portTransmitRate*** as defined in 8.6.8.2.
b) ***deltaBandwidth(N)***: The ~~additional~~ bandwidth, represented as a percentage of *portTransmitRate,* that can be reserved by SRP for use by the queue associated with traffic class N~~, in addition to the *deltaBandwidth(N)* values associated with higher priority queues. For a given traffic class N, the total bandwidth that can be reserved is the sum of the *deltaBandwidth* values for traffic class N and all higher traffic classes, minus any bandwidth reserved by higher traffic classes that support the credit-based shaper algorithm (see 34.3.1)~~. The interpretation of *deltaBandwidth(N)* is determined by the value of the associated *lockClassBandwidth(N)* as specified in 34.3.1 and 34.3.2.
c) ***adminIdleSlope(N)***: The bandwidth, in bits per second, that has been requested by management to be reserved for use by the queue associated with traffic class N. If SRP is in operation for traffic class N, this parameter has no effect~~; i~~. If SRP is not in operation for traffic class N, then the value of *operIdleSlope(N)* is always equal to the value of *adminIdleSlope(N)*.
d) ***operIdleSlope(N)***: The actual bandwidth, in bits per second, that is currently reserved for use by the queue associated with traffic class N (see 34.6.1 and 34.6.2)~~. This value is used by the credit based shaper algorithm (8.6.8.2) as the idleSlope for the corresponding queue.~~
e) ***classMeasurementInterval(N)***: The interval of time, in seconds, over which SRP's TSpec (34.4, 35.2.2.8.4) is measured for traffic class N at the Talker.
f) ***lockClassBandwidth(N)***: The Boolean parameter that determines the interpretation of *deltaBandwidth(N)*. For the value false (default), *deltaBandwidth(N)* is specified in 34.3.1. For the value true, *deltaBandwidth(N)* is specified in 34.3.2.

In all cases, bandwidth is defined in terms of the actual bandwidth consumed when frames are transmitted through the Port, not the size of the MSDU "payload" carried within those frames. Subclause 34.4 discusses the relationship between MSDU size and actual bandwidth consumed.

~~NOTE—While the deltaBandwidth values are specified with respect to specific traffic classes, and indicate the amount of bandwidth that can be reserved for traffic belonging to a particular traffic class, this does not imply that these traffic~~

~~classes have preferential access to that portion of the bandwidth. The priority of a given traffic class does not, for example, imply anything about the importance of a data stream that uses that class; the reservation strategy might therefore allocate bandwidth to a high importance stream that uses a lower priority traffic class in preference to a stream of lower importance that uses a higher priority traffic class.~~

34.3.1 ~~Relationships among bandwidth availability parameters~~deltaBandwidth when lockClassBandwidth is false

When *lockClassBandwidth(N)* is false, *deltaBandwidth(N)* is the additional bandwidth, represented as a percentage of *portTransmitRate*, that can be reserved by SRP for use by the queue associated with traffic class N, in addition to the *deltaBandwidth(N)* values associated with higher priority queues with *lockClassBandwidth(N)* false. For a given traffic class N, the total bandwidth that can be reserved is the sum of the *deltaBandwidth* values for traffic class N and all higher traffic classes with *lockClassBandwidth(N)* false, minus any bandwidth reserved by higher traffic classes (i.e., configured as SR classes).

The default value of *lockClassBandwidth(N)* is false. This default can be changed using the managed objects of the Bandwidth Availability Parameter Table (12.20.1)

The ~~recommended~~ default value of *deltaBandwidth(N)* for the highest numbered traffic class supported is 75%, and for any lower numbered traffic classes, the ~~recommended~~ default value is 0%. The *deltaBandwidth(N)* for a given *N*, plus the *deltaBandwidth(N)* values for any higher priority queues (larger values of *N*) defines the total percentage of the Port's bandwidth that can be reserved for that queue and all higher priority queues. For the highest priority queue, this means that the maximum value of *operIdleSlope(N)* is *deltaBandwidth(N)*% of *portTransmitRate*. However, if *operIdleSlope(N)* is actually less than this maximum value, any lower priority queue ~~that supports the credit-based shaper algorithm~~ with *lockClassBandwidth(N)* false can make use of the reservable bandwidth that is unused by the higher priority queue. So, for queue N-1, the maximum value of (*operIdleSlope(N)* + *operIdleSlope(N-1)*) is (*deltaBandwidth(N)* + *deltaBandwidth(N-1)*)% of *portTransmitRate*.

NOTE 1—For example, suppose two queues, 3 and 2, ~~support the credit-based shaper algorithm for~~ are configured as SR classes A and B, respectively. Suppose *deltaBandwidth(3)* for SR class A is currently 20%, and *deltaBandwidth(2)* for SR class B is currently 30%, and *lockClassBandwith* is false for both. If *operIdleSlope(3)* is currently 10% of *portTransmitRate*, then half of queue 3's maximum allocation is unused, and the maximum value of *operIdleSlope(2)* is therefore currently 40% of *portTransmitRate*. However, if *operIdleSlope(3)* increases to the full 20% that it is entitled to use, the maximum value of *operIdleSlope(2)* reduces to 30% of *portTransmitRate*.

NOTE 2—The sum of the *deltaBandwidth(N)* values for all values of N should be chosen such that there is sufficient bandwidth available for any nonreserved (best-effort, strict-priority) traffic; the default values are chosen such that the sum of the *deltaBandwidth(N)* values is 75%, so no more than 75% of the Port's available bandwidth is permitted to be reserved. This ensures that when using default settings, there is at least 25% of the Port's bandwidth available for nonreserved traffic. However, as these default settings may be inappropriate for some situations (e.g., links that offer very high bandwidth, or networks with very low levels of nonreserved traffic), they can be modified by management.

34.3.2 deltaBandwidth when lockClassBandwidth is true

When *lockClassBandwidth(N)* is true, *deltaBandwidth(N)* is the maximum bandwidth, represented as a percentage of *portTransmitRate*, that can be reserved by SRP for use by the queue associated with traffic class N. The bandwidth limit of *deltaBandwidth(N)* is not related to higher or lower priority traffic classes (i.e., it is not truly a delta).

NOTE—For example, suppose two queues, 3 and 2, are configured as SR classes A and B, respectively. Suppose *deltaBandwidth(3)* for SR class A is currently 20%, *deltaBandwidth(2)* for SR class B is currently 30%, and *lockClassBandwith* is true for both. If *operIdleSlope(3)* is currently 10% of *portTransmitRate*, then half of queue 3's maximum allocation is unused. Nevertheless, the maximum value of *operIdleSlope(2)* remains 30% of *portTransmitRate*. The value of *operIdleSlope(2)* cannot increase to 40%, because 20% is locked to use by *operIdleSlope(3)*.

34.3.3 ~~34.3.2~~ Bandwidth availability parameter management

The values of *deltaBandwidth(N)* and *adminIdleSlope(N)* can be changed by management, using the management operations defined in 12.20. If the stream reservation mechanisms defined in SRP are supported, then the values of *operIdleSlope(N)* are determined solely by the operation of SRP. If the stream reservation mechanisms defined in SRP are not supported, then the values of *operIdleSlope(N)* are equal to the values requested by management in the corresponding *adminIdleSlope(N)* parameters.

It is possible for the value of *portTransmitRate* for a Port to change as a result of the normal operation of the underlying MAC Service (e.g., as a result of the operation of IEEE 802.1AX Link Aggregation, or as a result of dynamic changes in bandwidth of the physical layer technology itself, as can occur in wireless LAN technologies); it is also possible for management action to change the values of *deltaBandwidth(N)* for a Port. In either case, the consequence could be one of the following:

a) The sum of the *operIdleSlope(N)* values for the Port could now exceed the total reservable bandwidth allowed for the Port, or the *operIdleSlope(N)* value for a given queue could now exceed the reservable bandwidth allowed for the queue, as defined in 34.3.1 and 34.3.2. Consequently, there could be streams currently active on the Port that can no longer be supported.

b) The bandwidth now available to a given queue could mean that there are streams that are currently inactive that could be supported on the Port.

c) Active streams that continue to be supported after the change could see their latency guarantee change.

In either case, corrective action, either by management or by the stream reservation mechanisms defined in SRP, is required in order to restore the parameters to a consistent set of values. In order to indicate that there have been changes in the bandwidth availability database, a signal, *bandwidthAvailabilityChanged*, is generated for each Port whenever the bandwidth availability parameters *portTransmitRate* or *deltaBandwidth(N)* are changed; this signal is used by SRP to trigger recalculation of the set of streams that can be reserved on the Port.

NOTE—During recalculation of stream reservations, the Bridge might be temporarily unable to honor bandwidth reservation commitments or forward best-effort traffic.

34.4 Deriving actual bandwidth requirements from the size of the MSDU

The forwarding and queuing mechanisms defined in this clause use bandwidth parameters that are defined in terms of the actual bandwidth used when frames are transmitted on the medium that supports the MAC Service available through the Port. In contrast, the SRP makes use of a traffic specification (TSpec) for each stream that defines the maximum number of bits per frame (*MaxFrameSize*), of the mac_service_data_unit parameter that is relayed by the relay function of the Bridge, and a maximum frame rate (*MaxIntervalFrames*), in frames per class measurement interval, for that stream; i.e., the TSpec takes no account of the per-frame overhead associated with transmitting the MSDU over a given medium. However, when SRP determines the value to be used for the *operIdleSlope(N)* parameter associated with a given queue, it is necessary for this value to include the per-frame overhead that will be incurred when frames are transmitted on that Port.

NOTE 1—The frame rate in a TSpec is measured over ~~a "class measurement interval"~~the classMeasurementInterval (34.3) that depends upon the SR class associated with the stream. Default values for classMeasurementInterval are specified in 34.5.~~SR class A corresponds to a class measurement interval of 125 µs; SR class B corresponds to a class measurement interval of 250 µs. These class measurement intervals apply at the source of the stream, i.e., the "Talker" end station, and do not necessarily hold good for subsequent stages in the stream's transmission across a Bridged Network.~~

For the purposes of calculating the bandwidth consumption of a stream, it is assumed that the stream data is essentially of constant size and transmission rate, so these maxima can be used to directly define an assumed maximum payload size and the maximum frame rate in frames per second; i.e.,

$$assumedPayloadSize = MaxFrameSize \quad (34\text{-}1)$$

$$maxFrameRate = MaxIntervalFrames \times (1/classMeasurementInterval) \quad (34\text{-}2)$$

where *classMeasurementInterval* is measured in seconds.

NOTE 2—As stated, the calculation of bandwidth from TSpec parameters assumes that the stream data is essentially of constant frame size, and hence, the approximations shown in this section are valid. If the data varies significantly in frame size, then the of per-frame overhead using these assumptions could be significantly in error.

From this, and also from local knowledge of the protocol stack that supports the Bridge Port, it is possible to determine the overhead that is added to the per-frame MSDU payload when a frame is transmitted. There are at least the following sources of per-frame overhead:

a) Any VLAN tags and security tags (see IEEE Std 802.1AE) that are added to the layer 2 payload as it passes through the various service interfaces in the Port's protocol stack.
b) The MAC framing (header and trailer octets, plus any padding octets that are required to meet minimum frame size limitations) that is added by the underlying MAC Service.
c) Any physical layer overhead, such as preamble characters and inter-frame gaps.

The precise per-frame overhead will therefore depend upon the protocol stack and the underlying MAC technology.

The actual bandwidth needed to support a given stream is therefore defined as follows [using *assumedPayloadSize* from Equation (34-1)]:

$$actualBandwidth = (perFrameOverhead + assumedPayloadSize) \times maxFrameRate \quad (34\text{-}3)$$

34.5 ~~Mapping priorities to traffic classes for time-sensitive streams~~Default SR class configuration

This subclause specifies the default configuration of SR classes for a Bridge or end station that supports FQTSS, including traffic class, priority, transmisssion selection algorithm, and classMeasurementInterval.

In Bridges that support FQTSS, the default mappings of priorities to traffic classes meet the following constraints:

a) Priority values that correspond to SR classes are mapped onto traffic classes that support the credit-based shaper algorithm as the transmission selection algorithm.
b) Traffic classes that support the credit-based shaper algorithm have a higher priority than traffic classes that support the strict priority (or any other) transmission selection algorithm.
c) At least one traffic class supports the credit-based shaper algorithm, and at least one traffic class supports the strict priority transmission selection algorithm.

NOTE 1—The constraint that there is at least one traffic class that supports the strict priority transmission selection ensures that there is at least one traffic class that can support traffic that is not subject to bandwidth reservation, such as "best effort" traffic.

The ~~recommended~~ default priority to traffic class mappings for a system that supports SR class A (using priority 3) and SR class B (using priority 2) are shown in Table 34-1. The ~~recommended~~ default priority to

traffic class mappings for a system that supports only SR class B (using priority 2) are shown in Table 34-2. These defaults can be changed using the managed objects of the Traffic Class Table for each Port (12.6.3).

The corresponding default configuration for the ~~Transmission Selection Algorithm Table~~ transmission selection algorithm (see 8.6.8) is that the traffic classes that are shaded in the tables are configured to use the credit-based shaper algorithm, and the remaining traffic classes are configured to use the strict priority algorithm. Traffic classes that are shaded correspond to default SR classes. These defaults can be changed using the managed objects of the Transmission Selection Algorithm Table (12.20.2).

Table 34-1—~~Recommended~~Default priority to traffic class mappings for SR classes A and B

		Number of available traffic classes						
		2	**3**	**4**	**5**	**6**	**7**	**8**
Priority	0 (Default)	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	0
	2	1	1	2	3	4	5	6
	3	1	2	3	4	5	6	7
	4	0	0	1	1	1	1	2
	5	0	0	1	1	1	2	3
	6	0	0	1	2	2	3	4
	7	0	0	1	2	3	4	5

Table 34-2—~~Recommended~~Default priority to traffic class mappings for SR class B only

		Number of available traffic classes						
		2	**3**	**4**	**5**	**6**	**7**	**8**
Priority	0 (Default)	0	0	0	0	0	1	1
	1	0	0	0	0	0	0	0
	2	1	2	3	4	5	6	7
	3	0	0	0	1	1	2	2
	4	0	1	1	2	2	3	3
	5	0	1	1	2	2	3	4
	6	0	1	2	3	3	4	5
	7	0	1	2	3	4	5	6

NOTE 2—The mapping shown in Table 34-1 for the case of only two available traffic classes maps two SR classes to a single traffic class. While this is a permissible configuration, in general it is desirable, and in some applications, can be a requirement, to assign SR classes to distinct traffic classes, as is done in this table for the cases where three or more traffic classes are available. The mappings shown deal only with one or two supported SR classes; a similar mapping strategy can be adopted if more than two SR classes are supported.

Table 34-1 and Table 34-2 specify a default configuration of SR class A as priority 3, and SR class B as priority 2. These defaults can be changed using the managed objects of the SR Class to Priority Mapping Table (12.20.4).

The default classMeasurementInterval (34.3) of SR class A is 125 µs. The default classMeasurementInterval of SR class B is 250 µs. These defaults can be changed using the managed objects of the Bandwidth Availability Parameter Table (12.20.1).

34.6 ~~End station behavior~~Transmission selection

Transmission selection (8.6.8) is used to shape the transmission of a Stream's frames in accordance with the bandwidth that has been reserved on a given outbound queue for a traffic class.

The following transmission selection algorithm shall be supported for FQTSS, configured as the default for SR class A and/or B:

a) Credit-based shaper algorithm, specified in 8.6.8.2

The following transmission selection algorithm may be supported for FQTSS, configured using the managed objects of the Transmission Selection Algorithm Table (12.20.2):

b) Strict priority algorithm, specified in 8.6.8.1

The following transmission technique may be supported for FQTSS, configured using the managed objects for scheduled traffic (12.29):

c) Scheduled traffic, specified in 8.6.8.4

34.6.1 Credit-based shaper

The *operIdleSlope(N)* parameter (34.3) is used by the credit-based shaper algorithm (8.6.8.2) as its idleSlope for the corresponding queue, to shape outbound traffic.

The *operIdleSlope(N)* parameter is also used to reserve space in the queue of the traffic class that is assigned to the SR class.

In order for an end station to successfully participate in the transmission and reception of time-sensitive streams, it is necessary for their behavior to be compatible with the operation of the forwarding and queuing mechanisms employed in Bridges. The requirements for end stations that participate as "Talkers"—i.e., sources of time-sensitive streams—are different from the requirements that apply to "Listeners"—i.e., the destination station(s) for the streams.

34.6.1.1 ~~34.6.1~~ Talker behavior

In order for Talker-originated data streams to make use of the credit-based shaper behavior in Bridges, it is a requirement for a Talker to use the priorities that the Bridges in the network recognize as being associated with SR classes exclusively for transmitting stream data. It is also necessary for the Talker, and the Bridges in the path to the Listener(s), to have a common view of the bandwidth required in order to transmit the

Talker's streams, and for that bandwidth to be reserved along the path to the Listener(s). This latter requirement can be met by means of stream reservation mechanisms, such as defined in SRP, or by other management means.

End stations that are Talkers shall exhibit transmission behavior for frames that are part of time-sensitive streams that is consistent with the operation of the credit-based shaper algorithm, both in terms of the way they transmit frames that are part of an individual data stream, and in terms of the way they transmit stream data frames from a Port. In effect, the queuing model for a Talker Port, and for a given priority, can be considered to look like Figure 34-1.

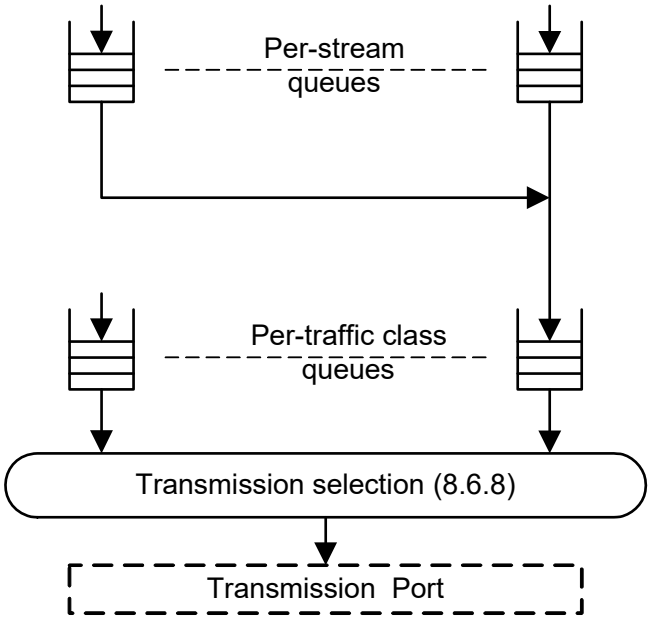

Figure 34-1—Queuing model for a Talker station

The Talker places frames into the queue associated with an individual stream based on the Tspec for that stream; i.e., during each ~~class measurement interval~~ *classMeasurementInterval*, it can place up to *MaxIntervalFrames* data frames, each no longer than *MaxFrameSize* into that stream's queue.

The queue associated with each individual stream uses the credit-based shaper algorithm, with the *idleSlope* set to the bandwidth requirement of the stream on that transmission Port, as the means of determining the rate at which data frames for that stream are placed in the outbound queue for the priority that the stream is using. The outbound queue for that priority, in turn, makes use of the credit-based shaper algorithm, with the *idleSlope* set to the sum of the *idleSlope* values for all streams using that priority on that transmission Port, as the means of determining the rate at which data frames for that stream are selected for transmission.

Transmission selection in a Talker operates in the same way as in a Bridge; from this point of view, a Talker can be thought of as if it is a single-port Bridge.

For streams that make use of SR class A or SR class B, it is a requirement that the rate at which frames for any given stream are selected for placement in its per-stream queue does not exceed the bandwidth reserved for the stream, measured over the ~~class measurement interval~~ *classMeasurementInterval* (34.3) for the SR class ~~(125 µs for SR class A, 250 µs for SR class B.)~~. For some combinations of stream bandwidth requirement and transmission Port data rate, this can place a limit on the frame size that can be used when transmitting stream data.

NOTE—The sole implication of the ~~observation interval~~ *classMeasurementInterval* is its effect on frame size, because the shaper behavior itself is independent of the ~~observation interval~~ *classMeasurementInterval*. The intent in limiting the ~~observation interval~~ *classMeasurementInterval* is to limit frame size, as this is a major contribution to the latency experienced by a frame in transit through an ~~audio/video bridging (AVB)~~ time-sensitive network (see discussion in Annex L).

34.6.1.2 ~~34.6.2~~ Listener behavior

The primary requirement for a Listener end station using the credit-based shaper is that it is capable of buffering the amount of data that could be transmitted for a stream during a time period equivalent to the accumulated maximum jitter that could be experienced by stream data frames in transmission between Talker and Listener. From the point of view of the specification of the forwarding and queuing requirements for time-sensitive streams, it is assumed that the Listener will assess the buffering required for a stream as part of the stream bandwidth reservation mechanisms employed by the implementation.

34.6.2 Strict priority

The *operIdleSlope(N)* parameter (34.3) represents the total outbound bandwidth for the queue of traffic class N, and *operIdleSlope(N)* shall be used to reserve space in the queue of the associated traffic class N.

NOTE 1—The *operIdleSlope(N)* parameter is not used by the algorithm for transmission selection (8.6.8.1).

When SRP is used for reservation of a stream, the egress Port that uses strict priority will place frames into its outbound queue based on the Tspec for that Stream. The TSpecs of all Streams that egress the Port are used to compute the *operIdleSlope(N)*.

When SRP is not used, the worst-case bandwidth for traffic class N is computed by a management client (e.g., CNC of 46.1.3.3, using TSpecs from the CUC) for the queue of the associated traffic class N. The management client uses *adminIdleSlope(N)* to configure this bandwidth in the station (12.20.1). Since *operIdleSlope(N)* is equal to *adminIdleSlope(N)*, *operIdleSlope(N)* is used to reserve space in the queue of the associated traffic class N. Since the priority associated with traffic class N is not using SRP, that priority does not need to be configured as an SR class (i.e., assigned an SRclassID using 12.20.4).

NOTE 2—As an example of a traffic class configured using *adminIdleSlope(N)*, consider a traffic class that is protected using scheduled traffic (8.6.8.4). This protected traffic class can use the strict priority algorithm. Traffic in the protected window requires queue reservation in order to ensure that adequate storage exists in its queue. The CNC entity of 46.1.3.3 can use *adminIdleSlope(N)* to ensure that the queue for the protected traffic class is of sufficient size.

NOTE 3—Although *adminIdleSlope(N)* is specified as bits per second, with regard to reserving queue space for a shorter interval of time, it is best to assume that *adminIdleSlope(N)* is uniform over the entire second. For example, for a traffic class that is protected using scheduled traffic (8.6.8.4), if AdminCycleTime is 1 ms, *adminIdleSlope(N)* is divided by 1000 to determine the worst-case traffic for that AdminCycleTime.

34.6.3 Scheduled traffic

When SRP uses Centralized Network Configuration (CNC), scheduled traffic (8.6.8.4) can be used in a Talker as well as the Bridges to each Listener. For information, refer to the specifications for externalControl TRUE in 35.2.2.10.

When SRP is fully distributed (externalControl FALSE in 35.2.2.10), use of scheduled traffic in Bridges is not specified for SRP's Streams. If scheduled traffic is used in a Talker, that Talker appends the TSpec-TimeAware TLV to its Talker Advertise. If the nearest Bridge supports the optional capability of translating scheduled traffic in the Talker to the transmission selection used in Bridges (i.e., 34.6.1 or 34.6.2), the nearest Bridge propagates Talker Advertise. As specified in 35.2.2.10.6, if the nearest Bridge does not support the TSpecTimeAware TLV, the nearest Bridge fails the Stream (propagates Talker Failed).

NOTE—For an example using scheduled traffic in a Talker, refer to U.1.1.

35. Stream Reservation Protocol (SRP)

Change the first three paragraphs of the introductory text of Clause 35 as shown:

SRP utilizes three signaling protocols~~,~~ [MMRP (10.9), MVRP (Clause 11), and MSRP (35.1)~~,~~] to establish stream reservations across a bridged network.

Within SRP the Multiple MAC Registration Protocol (MMRP) ~~is optionally~~ may be used to control the propagation of Talker registrations throughout the bridged network (~~35.2.4.3.1~~35.2.4.3.2).

The Multiple VLAN Registration Protocol (MVRP) is used by end stations and Bridges to declare membership in a VLAN where a Stream is being sourced. This allows the Data Frame Priority [item a) in 35.2.2.8.5] to be propagated along the path from Talker to Listener(s) in tagged frames. MSRP will not allow Streams to be established across Bridge Ports that are members of the untagged set (8.8.10) for the related VLAN ID.

Insert the following paragraph at the end of the introductory text of Clause 35:

The preceding description of SRP as a signaling protocol corresponds to the Fully distributed model (46.1.3.1) for TSN configuration. SRP also supports the Centralized network/distributed user model (46.1.3.2) for TSN configuration, which can be used to enable enhanced features (see 35.2.2.10). In order to use the Centralized network/distributed user model, the MRP External Control (12.32.4) feature is enabled in the Nearest Bridge (8.6.3) to each Talker and Listener. Instead of propagating (signaling) through Bridges, SRP messages are exchanged with a Centralized Network Configuration (CNC) entity. The CNC uses remote management protocols (e.g., SNMP, NETCONF) in order to configure Bridges to meet the Stream requirements of each Talker and Listener.

35.1 Multiple Stream Registration Protocol (MSRP)

Insert the following paragraphs at the end of the introductory text of 35.1:

The version of MSRP is distinguished by the ProtocolVersion of MRP. ProtocolVersion 0x00 (MSRPv0) is the original version. ProtocolVersion 0x01 (MSRPv1) or higher specifies new AttributeTypes (35.2.2.4) that provide enhanced capabilities. This standard specifies MSRPv1 (35.2.2.3). In order to meet interoperability goals, implementations of this version shall support the original AttributeTypes specified in MSRPv0 as well as the enhanced AttributeTypes. The interoperability goals require a network of mixed MSRPv0 and MSRPv1 end stations and Bridges to provide a quality of service that is as good as or better than a complete MSRPv0 network.

For both MSRPv0 and MSRPv1, each MSRP Participant declares the Domain attribute prior to other MSRP attributes in order to determine SR class boundaries. The MSRPv1 end station or Bridge also uses this Domain exchange to determine the ProtocolVersion of its neighboring MSRP Participant. If the neighbor is MSRPv0, then only the original AttributeTypes are exchanged with that neighbor, although the ProtocolVersion will be set to the value defined in 35.2.2.3 [see item b) in 10.8.3.5]. If the neighbor is MSRPv1, then both original and enhanced AttributeTypes can be exchanged with that neighbor. For a Bridge, this means that it is possible to register an enhanced AttributeType from one Port and propagate and declare that same attribute to another Port as an original AttributeType. To enable this mixture of AtributeTypes, the MSRP Attribute Propagation (35.2.4) specifies the translation from each enhanced AttributeType to its corresponding original AttributeType.

For propagation of an MSRP attribute from one Port using MSRPv1 to another Port using MSRPv1, the AttributeType is not changed. This allows original AttributeTypes to propagate through a Bridged Network as is, without translation to enhanced AttributeTypes.

NOTE—MRP (Clause 10) has limitations on the total amount of attribute data that can be exchanged between participants. Therefore, the larger each MSRP attribute's value, the fewer total Streams can be reserved in the network. Due to the enhanced features, the enhanced AttributeType is larger than the original AttributeType. If the Talker and Listeners of a Stream do not need the enhanced features, use of the original AttributeTypes can allow for a larger number of total Streams. If all Talkers and Listeners in the network use original AttributeTypes, the maximum number of Streams in an all-MSRPv1 network is the same as an all-MSRPv0 network.

35.1.2 Behavior of end stations

35.1.2.1Talkers

Insert the following paragraph and note after the first paragraph ("To announce ....") of 35.1.2.1:

The Talker shall issue an MVRP VLAN membership declaration prior to issuing the Talker Declaration. The MVRP VLAN membership shall contain the vlan_identifier of the DataFrameParameters [item b) in 35.2.2.8.3], so that the neighboring Bridge adds the associated Bridge Port to the member set for the VLAN.

NOTE—VLAN membership is needed for the Talker when the neighboring Bridge has the Enable Ingress Filtering parameter set (8.6.2).

35.1.2.2 Listeners

Change the note and the third, fourth, and fifth paragraphs of 35.1.2.2, and insert the additional paragraphs as shown:

NOTE 1—The reader ... as appropriate.

~~When there is a Talker Declaration registered on an interested Listener end station, the Listener shall create a Listener Declaration as follows:~~

If the Listener receives a Talker Advertise declaration and the Listener is ready to receive the Stream, the Listener shall ~~declare the following in the order specified:~~ issue an MSRP Listener Ready declaration for the Stream.

~~d) An MVRP VLAN membership request for the vlan_identifier contained in the Talker Advertise DataFrameParameters [item b) in 35.2.2.8.3] so the neighboring bridge will add the associated Bridge Port to the member set for the VLAN;~~
~~e) An MSRP Listener Ready declaration for the Stream.~~

The Listener shall issue an MVRP VLAN membership declaration prior to issuing the MSRP Listener Ready declaration. The MVRP VLAN membership shall contain the vlan_identifier of the Talker Advertise DataFrameParameters [item b) in 35.2.2.8.3], so that the neighboring Bridge adds the associated Bridge Port to the member set for the VLAN.

If the *talkerVlanPruning* parameter (35.2.4.3.4) is not enabled in the Bridges in the path from Talker to the Listener, the Listener receives Talker Advertise prior to declaring MVRP VLAN membership. The Listener can then use the registered Talker Advertise to obtain the vlan_identifier for declaration of MVRP VLAN membership. This is the default behavior for the *talkerVlanPruning* parameter.

If one or more Bridges in the path from Talker to the Listener have enabled the *talkerVlanPruning* parameter, the Listener declares VLAN membership prior to receiving the Talker Advertise declaration. The *talkerVlanPruning* parameter instructs the Bridge to filter Talker declarations for the VLAN (in addition to the VLAN-tagged data frames of the stream). If the *talkerVlanPruning* parameter is enabled and the VLAN is not Registration Fixed (8.8.10), the Listener obtains knowledge of the Talker's vlan_identifier from a

higher layer protocol above IEEE Std 802.1Q and uses that knowledge to declare the membership for the vlan_identifier with MVRP.

If *talkerPruning* or *talkerPruningPerPort* is enabled, the Listener must issue an MMRP registration for the Stream's destination MAC address as described in 35.2.4.2 before the Talker Advertise will be received.

If the Listener receives a Talker Failed declaration, and the Listener is ready to receive the Stream, the Listener shall issue ~~a~~ either an MSRP Listener Asking Failed declaration or an MSRP Listener Ready declaration for the Stream.

NOTE 2—One reason for the Listener to issue Listener Ready in response to Talker Failed is that the Listener wants to inform the network that it is ready to proceed in case the Talker's state changes from failed to ready in the future.

35.1.3 Behavior of Bridges

Change the second paragraph as shown:

In general, Talker declarations are propagated to all other Bridge Ports. There ~~is a~~ are *talkerPruning* ~~option~~ [item b) in 35.2.1.4], *talkerPruningPerPort* [item k) in 35.2.1.4], and *talkerVlanPruning* [item l) in 35.2.1.4] parameters that limit~~s~~ the scope of Talker declaration propagation. Listener declarations are propagated only to the Bridge Port with the associated Talker declaration (i.e., matching StreamID). If there is no associated Talker declaration registered on any Bridge Port, then Listener declaration will not be propagated.

35.1.3.1 Blocked Declarations

Change 35.1.3.1 as shown:

For the purposes of MSRP Attribute Propagation (35.2.4), a Declaration is said to be "blocked" on a Bridge Port if the state of the Spanning Tree Instance identified by the vlan_identifier in the DataFrameParameters (35.2.2.8.3) of the Declaration, on that Bridge Port, has any value other than Forwarding. In an end station's MSRP Participant, no Declaration is ever blocked.

35.2 Definition of the MSRP application

35.2.1 Definition of internal state variables

35.2.1.2 Direction

Change the lettered list in 35.2.1.2 as shown:

a) **Talker:** MSRP AttributeType definitions of type Talker Advertise Vector Attribute Type [item a) in 35.2.2.4] ~~or~~, Talker Failed Vector Attribute Type [item b) in 35.2.2.4], or Talker Enhanced Vector Attribute Type [item e) in 35.2.2.4]. Set Direction to zero for Talker attributes.
b) **Listener:** MSRP AttributeType definitions of type Listener Vector Attribute Type [item c) in 35.2.2.4] or Listener Enhanced Vector Attribute Type [item f) in 35.2.2.4]. Set Direction to one for Listener attributes.

35.2.1.3 Declaration Type

Change the lettered list in 35.2.1.3 as shown:

For a Talker, the value of the Declaration Type component is either

a) **Advertise:** MSRP AttributeType definitions of Talker Advertise Vector Attribute Type [item a) in 35.2.2.4] or Talker Enhanced Vector Attribute Type [item e) in 35.2.2.4] with Status.StatusInfo.TalkerStatus equal to Ready (35.2.2.10.9). Set Declaration Type to zero for Talker Advertise.
b) **Failed:** MSRP AttributeType definitions of Talker Failed Vector Attribute Type [item b) in 35.2.2.4] or Talker Enhanced Vector Attribute Type [item e) in 35.2.2.4] with Status.StatusInfo.TalkerStatus equal to Failed (35.2.2.10.9). Set Declaration Type to one for Talker Failed.

For a Listener, the value of the Declaration Type component is one of the following:

c) **Asking Failed:** MSRP AttributeType definitions of Listener Vector Attribute Type [item c) in 35.2.2.4] with MSRP FourPackedType equal to Asking Failed [item b) in 35.2.2.7.2] or Listener Enhanced Vector Attribute Type [item f) in 35.2.2.4] with Status.StatusInfo.ListenerStatus equal to Failed (35.2.2.10.9). Set Declaration Type to two for Listener Asking Failed.
d) **Ready:** MSRP AttributeType definitions of Listener Vector Attribute Type with MSRP FourPackedType equal to Ready [item c) in 35.2.2.7.2] or Listener Enhanced Vector Attribute Type [item f) in 35.2.2.4] with Status.StatusInfo.ListenerStatus equal to Ready (35.2.2.10.9). Set Declaration Type to three for Listener Ready.
e) **Ready Failed:** MSRP AttributeType definitions of Listener Vector Attribute Type with MSRP FourPackedType equal to Ready Failed [item d) in 35.2.2.7.2] or Listener Enhanced Vector Attribute Type [item f) in 35.2.2.4]) with Status.StatusInfo.ListenerStatus equal to PartialFailed (35.2.2.10.9). Set Declaration Type to four for Listener Ready Failed.

35.2.1.4 SRP parameters

Change item b) in the lettered list in 35.2.1.4 as shown:

b) ***talkerPruning*:** Enabling this parameter on the Bridge will limit the Talker declarations to ~~p~~Ports that have the Stream's destination_address [item a) in 35.2.2.8.3] in the ~~MMRP~~ MAC Address Registration Entries (8.8.4), such as by using MMRP.

Insert the following items at the end of the lettered list in 35.2.1.4:

j) ***neighborProtocolVersion*:** Each Port provides a neighborProtocolVersion parameter that specifies the MSRP ProtocolVersion of the neighboring MSRP Participant on that Port. The initial value is 0x00. When an MSRP Domain attribute is registered (received) on the Port, the value of the attribute's MRP ProtocolVersion (35.2.2.3) is copied to this parameter.
k) ***talkerPruningPerPort*:** When the talkerPruning parameter is disabled (false), each Port provides a talkerPruningPerPort parameter to control MAC address pruning for that Port. For more information, refer to 35.2.4.3.3.
l) ***talkerVlanPruning*:** Enabling this parameter on the Bridge will limit the Talker declarations to Ports that have the Stream's vlan_identifier [item b) in 35.2.2.8.3] registered as a member in the VLAN Registration Entries (8.8). For more information, refer to 35.2.4.3.4.
m) ***maxSRclasses*:** This parameter provides the maximum number of SR classes supported by the Bridge.

35.2.2 Definition of MRP elements

35.2.2.3 MSRP ProtocolVersion

Change 35.2.2.3 as shown:

The ProtocolVersion for the version of MSRP defined in this standard takes the hexadecimal value ~~0x00~~ 0x01.

NOTE—Although an MSRP Participant for this standard transmits MSRPDUs using ProtocolVersion 0x01, the MSRP Participant must process received MSRPDUs using ProtocolVersion 0x01 or 0x00, as specified in 10.8.3.5.

35.2.2.4 MSRP AttributeType definitions

Change 35.2.2.4, including Table 35-1, as shown:

MSRP defines ~~four~~ six AttributeTypes (10.8.2.2) that are carried in MRP exchanges. The numeric values for the AttributeType are shown in Table 35-1, and their use is defined by the following list:

a) **Talker Advertise Vector Attribute Type:** Attributes identified by the Talker Advertise Vector Attribute Type are instances of VectorAttributes (10.8.1), used to identify a sequence of values of Talker advertisements for related Streams that have not been constrained by insufficient bandwidth or resources. This Attribute Type applies to all versions of MSRP.
b) **Talker Failed Vector Attribute Type:** Attributes identified by the Talker Failed Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values of Talker advertisements for related Streams that have been constrained by insufficient bandwidth or resources. This Attribute Type applies to all versions of MSRP.
c) **Listener Vector Attribute Type:** Attributes identified by the Listener Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values of Listener requests for related Streams regardless of bandwidth constraints. Listener Vector Attribute Types are subdivided into individual Declaration Types via the MSRP FourPackedEvents (35.2.2.7.2). This Attribute Type applies to all versions of MSRP.
d) **Domain Vector Attribute Type:** Attributes identified by the Domain Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values that describe the characteristics of an SR class. This Attribute Type applies to all versions of MSRP.
e) **Talker Enhanced Vector Attribute Type:** Attributes identified by the Talker Enhanced Vector Attribute Type are instances of VectorAttributes (10.8.1), used to identify a sequence of values of Talker advertisements for related Streams. Talker Enhanced Vector Attribute Types are subdivided into individual Declaration Types (35.2.1.3) via the Status.StatusInfo.TalkerStatus (35.2.2.10.9). This Attribute Type applies to ProtocolVersion 0x01 or higher.
f) **Listener Enhanced Vector Attribute Type:** Attributes identified by the Listener Enhanced Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values of Listener requests for related Streams. Listener Enhanced Vector Attribute Types are subdivided into individual Declaration Types (35.2.1.3) via the Status.StatusInfo.ListenerStatus (35.2.2.10.9). This Attribute Type applies to ProtocolVersion 0x01 or higher.

Table 35-1—AttributeType Values

AttributeType	Value	Used when neighborProtocolVersion is 0x00	Used when neighborProtocolVersion is 0x01 or higher
Talker Advertise Vector	1	yes	yes
Talker Failed Vector	2	yes	yes
Listener Vector	3	yes	yes
Domain Vector	4	yes	yes
Talker Enhanced Vector	5	no	yes
Listener Enhanced Vector	6	no	yes

35.2.2.5 MSRP AttributeLength definitions

Change Table 35-2 as shown:

Table 35-2—AttributeLength Values

AttributeType	Value
Talker Advertise Vector	25 (0x19)
Talker Failed Vector	34 (0x22)
Listener Vector	8
Domain Vector	4
Talker Enhanced Vector	variable
Listener Enhanced Vector	variable

Insert the following paragraph at the end of 35.2.2.5:

The AttributeLength field in instances of the Talker Enhanced Vector Attribute Type and Listener Enhanced Vector Attribute Type shall be encoded in MRPDUs as an unsigned binary number. The value of AttributeLength varies according to the TLVs contained in the attribute's value, but shall not change during the duration of the attribute's declaration.

35.2.2.7 MSRP Vector definitions

35.2.2.7.2 MSRP FourPackedEvents

Insert NOTE 2 after the now NOTE 1 in 35.2.2.7.2 as shown:

NOTE 1—In terms of efficient use of octets within an MSRP packet, placing nineteen (19) Ignores between two Listener declarations uses less octets than using two VectorAttributes to declare the Listener attributes separately. A single Listener VectorAttribute takes 12 octets, two attributes would take 24 octets. Declaring 21 attributes (two valid attributes with 19 Ignores in between) takes 12 octets, plus 6 additional ThreePackedEvents, plus 5 additional FourPackedEvents for a total of 23 octets.

NOTE 2—MSRP FourPackedEvents are not used for the Listener Enhanced Vector Attribute Type [item f) in 35.2.2.4], which uses the Status.StatusInfo.ListenerStatus (35.2.2.10.9) to distinguish Declaration Types (35.2.1.3).

Change the title and first paragraph of 35.2.2.8, and insert a note after the first paragraph as shown:

35.2.2.8 MSRP FirstValue definitions (Stream reservations, original)

There are four Attribute Declarations defined for the original ProtocolVersion 0x00 of MSRP (35.2.2.4), three of which are related to stream reservations: Talker Advertise, Talker Failed, and Listener. The fourth attribute type, Domain, is used to discover the SRP domain, as well as the ProtocolVersion of each Port's neighbor, and is described in 35.2.2.9.

NOTE—This version of MSRP requires implementation of the original attributes of 35.2.2.8 and the enhanced attributes of 35.2.2.10. The specifications for enhanced attributes are based on the specifications for original attributes.

35.2.2.8.2 StreamID

Change the lettered list in 35.2.2.8.2 as shown:

a) ~~An EUI-48~~ A 48-bit MAC Address associated with the ~~System~~ Talker sourcing the stream to the bridged network. The entire range of ~~EUI-48~~ MAC addresses are acceptable.
b) A 16-bit unsigned integer value, Unique ID, used to distinguish among multiple streams sourced by the same ~~System~~ Talker.

Change NOTE 2 and insert NOTE 3 in 35.2.2.8.2 as shown:

NOTE 2—The MAC address component of the StreamID can, but does not necessarily, have the same value as the source_address parameter of any frame in the actual data stream. For example, the StreamID can be assigned by a TSN CUC (see 46.1.3.3), using a pool of MAC addresses that the TSN CUC maintains.

NOTE 3—If the MAC addresses used to construct StreamIDs are not unique within the network, duplicate StreamIDs can be generated, with unpredictable results for SRP.

35.2.2.8.3 DataFrameParameters

Change the second paragraph of 35.2.2.8.3 as shown:

The destination_address specifies the destination MAC address of the streaming data packets. Only one Stream ~~Talker~~ is allowed per destination_address. MSRP does not describe the actual streaming data, only the bandwidth associated with that stream.

35.2.2.8.4 TSpec

Change item b) in the lettered list in 35.2.2.8.4 as shown:

b) **MaxIntervalFrames:** The 16-bit unsigned MaxIntervalFrames component is used to allocate resources and adjust queue selection parameters in order to supply the QoS requested by an MSRP Talker Declaration. It represents the maximum number of frames that the Talker may transmit in one ~~"class measurement interval" (34.4)~~ classMeasurementInterval (34.3).

35.2.2.8.6 Accumulated Latency

Change the sixth paragraph of 35.2.2.8.6 as shown:

For item d), the propagation time, if the managed object for Propagation Delay (12.32.2) is supported, txPropagationDelay shall be used. Otherwise, in the absence of better information, a value of 500 ns shall be used.

35.2.2.8.7 FailureInformation

Change 35.2.2.8.7, including deleting Table 35-6, as shown:

At the point when a Talker Advertise Declaration is transformed into a Talker Failed Declaration, the system making the transformation adds information that indicates, to the Listeners registering the Talker Failed Declaration, the cause of the failure and the identity of the Bridge at which the failure occurred. The subcomponents of the FailureInformation include

a) The system identifier, which is the Bridge Identifier (13.26.2) of the Bridge or the 48-bit MAC address of the end station's port extended to 64 bits by prepending 16 bits of zero, that changed the Declaration Type from Advertise to Failed.

NOTE 1—Bridge Identifiers are normally constructed from MAC Addresses that are unique in the bridged LAN but are not required to be constructed in that manner; therefore, there is a possibility of an end station MAC Address colliding with the Bridge ID.

b) The ~~Reservation~~ Failure Code, which is represented by a single octet containing the value shown in ~~Table 35-6~~ Table 46-15.

NOTE 2—Although the table of failure codes (Table 46-15) is located in a subclause associated with MSRPv1, the codes are compatible with MSRPv0.

NOTE 3—Table 35-6 is deleted.

~~**Table 35-6—Reservation Failure Codes**~~

~~Failure Code~~	~~Description of cause~~
~~1~~	~~Insufficient bandwidth~~
~~...~~	~~...~~
~~19~~	~~SR class priority mismatch~~

~~a A device ... AVB capable.~~
~~b This Failure Code ... not declared.~~

35.2.2.9 MSRP FirstValue definitions (Domain discovery)

Change 35.2.2.9 as shown:

The Domain attribute contains all the information that a Bridge Port needs in order to determine the location of the SRP domain boundary [item h) in 35.2.1.4]. The Domain attribute is also used to detect the ProtocolVersion of the neighboring MSRP Participant on each Bridge Port [item j) in 35.2.1.4].

The MSRP Participant shall declare (transmit) its Domain attribute after MAC_Operational (IEEE Std 802.1AC) transitions to TRUE and prior to declaring a Talker Advertise, Talker Failed, or Listener attribute.

FirstValue shall be incremented one or more times when NumberOfValues is greater than one. Incrementing FirstValue for the MSRP Domain attribute is defined as follows:

a) Add 1 to SRclassID (35.2.2.9.2), and
b) Add 1 to SRclassPriority (35.2.2.9.3).

The choice of encoding and incrementing with this rule means that if one class (e.g., class B) is supported, with the default values, then the FirstValue will be {5,2,VID} (class B, priority 2, and a VID) and the NumberOfValues field will be set to 1. If class A and class B are supported, with the default values, the FirstValue will again be {5,2,VID}, but the NumberOfValues fields will be set to 2. Applying the above incrementing rule to {5,2,VID} generates the value {6,3,VID}, i.e., class A, priority 3, and a VID, which is needed for the default case.

NOTE—If the SR Class to Priority Mapping Table (12.20.4) is configured through management such that SR class ID values do not increment contiguously with priority values, the declared domains need to be encoded in multiple Domain Vector Attributes. For example, if the SR class ID and priority rows are {2,4}, {3,0}, {4,1}, {5,2}, and {6,3}, two Domain attributes are needed: one Domain with FirstValue {2,4,VID} and NumberOfValues one, and a second Domain with FirstValue {3,0,VID} and NumberOfValues four.

35.2.2.9.2 SRclassID

Change 35.2.2.9.2 as shown:

SRclassID is a numeric representation of the SR classes that are supported by a particular Bridge Port. The mapping ~~for the first two SR classes~~ of SR class letter to SR class ID is shown in Table 35-7.

Table 35-7—SR class ID

SR class	SR class ID
A	6
B	5
C	4
D	3
E	2
F	1
G	0

NOTE—When SRP is supported, only one SR class is required. Only SR classes A and B can be enabled by default. Additional SR classes must be configured through use of management (12.20.4, 34.5).

35.2.2.9.3 SRclassPriority

Change 35.2.2.9.3 as shown:

This field holds the Data Frame Priority [item a) in 35.2.2.8.5] value that will be used for streams that belong to the associated SR class. Whenever an end station's MAC_Operational (IEEE Std 802.1AC)

transitions to TRUE, its SRclassPriority shall be set to the default SR class priority (Table 6-5) until an SRclassPriority declaration is received from a neighboring device, at which time it shall be set to the declared value. An end station cannot change its default mapping of SR class to priority (see 12.20.4).

NOTE—This allows two end stations that are connected back-to-back to share streams, while also providing an end station the flexibility to change to the SRclassPriority of an attached network (i.e., neighboring Bridge).

Insert the following subclause (35.2.2.10, including Table 35-8 and Figure 35-4 through Figure 35-17) after 35.2.2.9.4, and renumber the subsequent tables in Clause 35 accordingly:

35.2.2.10 MSRP FirstValue definitions (Stream reservations, enhanced)

This subclause specifies two Attribute Types for enhanced stream reservations: Talker Enhanced and Listener Enhanced.

The specifications of the enhanced Attribute Types (35.2.2.10) are based on the specifications of the original Attribute Types (35.2.2.8), with differences stated explicitly.

As compared to the original Attribute Types, the enhanced Attribute Types use a Type-Length-Value (TLV) encoding for MSRP FirstValue. MSRP FirstValue contains a list of TLVs. Each TLV consists of a Type field that specifies what the Value field contains, a Length field that specifies the number of octets in the Value field, and the Value field. The Value contains one or more elements encoded in binary. The TLV encoding enables flexibility in the structure of MSRP FirstValue, such as support for optional features.

This subclause specifies the structure of MSRP FirstValue as well as the Type and Length for each TLV. This subclause also specifies the binary encoding of the elements within the Value of each TLV. The semantic specification of each element in the TLV's Value is provided as a reference to the corresponding group of elements in TSN User/network configuration information (46.2).

The transmission of octets in MSRP FirstValue is specified in 10.8.1.1. The encoding of the Value of each TLV is specified in a figure, and the representation in the figure is specified in 10.8.1.1. Elements marked as "reserved" shall be transmitted as zero and ignored on reception. When an element is not a multiple of 8 bits in length, the element is organized from most significant bit of the octet to least significant bit. Boolean elements are a single bit. When an element's Length in the figure is a decimal number only, Length specifies the number of octets for the element. When an element's Length in the figure is a decimal number followed by "bits," Length specifies the number of bits for the element.

FirstValue shall be incremented one or more times when NumberOfValues is greater than one, as follows:

a) Add 1 to Unique ID of the StreamID TLV (46.2.3.1), and
b) Add 1 to each element of the DataFrameSpecification TLV (46.2.3.4) that identifies the Stream:
    1) If the IEEE802-MacAddresses TLV is used, add 1 to DestinationMacAddress.
    2) If the IPv4-tuple TLV is used, add 1 to DestinationIpAddress.
    3) If the IPv6-tuple TLV is used, add 1 to DestinationIpAddress.

This rule for incrementing FirstValue is the same as the original Attribute Types (35.2.2.8), with the addition of optional support for IP destination address as the unique identification of the Stream.

NOTE 1—For an example of incrementing FirstValue for DestinationMacAddress, refer to 35.2.2.8.

NOTE 2—As compared to the fixed attribute values of the original AttributeTypes, the use of TLVs in the enhanced AttributeTypes allows each attribute value to vary based on the use of sub-TLVs. This imposes an additional constraint when NumberOfValues is greater than one such that all FirstValues must use the same sub-TLVs.

NOTE 3—As specified in 35.2.3.1, when a Talker or Listener has a *neighborProtocolVersion* of 0x00 (i.e., Nearest Bridge is MSRPv0), declarations must use only the original Attribute Types (35.2.2.8). If the application of the Talker/Listener is using the enhanced features of this subclause (35.2.2.10), an internal translation is needed in order to determine if the original Attribute Type can be declared. This internal translation is similar to the translation specified for MAP in a Bridge (Table 35-12 and Table 35-17), but it is outside the scope of this standard. If the enhanced features do not translate successfully to an original Attribute Type, the Talker or Listener can choose to either a) declare the failure to the network using an original Attribute Type (e.g., Talker Failed Vector of Table 35-1) or b) return an error up to the application of the Talker/Listener.

Some of the enhanced features of this subclause require the assistance of a Centralized Network Configuration (CNC) entity. As described in the introduction to Clause 35, in order to use a CNC with SRP, the MRP External Control (12.32.4) feature is enabled in the Nearest Bridge (8.6.3) to each Talker and Listener. Therefore, when a Talker or Listener uses an enhanced feature, the success or failure of the Stream reservation depends on the externalControl (12.32.4.1) parameter on the Port connected to the Talker/Listener, as follows:

a) **externalControl TRUE**

SRP messages are exchanged with the CNC, and the CNC uses those SRP messages to learn the requirements for each Stream. The CNC uses remote management protocols (e.g., SNMP, NETCONF) in order to learn the capabilities of the Bridges between each Talker and its Listeners. The CNC is not required to support all Bridge features, but for the Bridge features that it supports, it compares the Bridge's capabilities to the configuration needed in order to meet each Stream's requirements. If the CNC successfully configures the Bridges for the Stream, the CNC shall use MRP External Control to declare a Talker Advertise attribute to each Listener and a Listener Ready attribute to the Talker. If the CNC fails to configure the Bridges for the Stream, the CNC shall use MRP External Control to declare a Talker Failed attribute to each Listener and a Listener Ready Failed or Listener Asking Failed attribute to the Talker.

b) **externalControl FALSE**

When MRP External Control is not used, SRP messages propagate through Bridges, and the Bridges configure themselves for the Stream reservation. Some enhanced features can be supported with this model, and other enhanced features cannot be supported. In the specifications of this subclause, the phrase "When the externalControl attribute is FALSE" is used to specify failure conditions when MRP External Control is not used for the Talker/Listener. If this phrase is not used, the feature is fully supported in the absence of MRP External Control (i.e., CNC).

35.2.2.10.1 Structure definition

The FirstValue of each attribute shall consist of a list of one or more TLVs. Each TLV shall contain a single octet TLV type, followed by a single octet TLV length, followed by multiple octets for the TLV value.

The TLV type shall use a value specified in Table 35-8.

Table 35-8—TLV types

TLV	TLV type	TLV length
Talker	1	variable
StreamID	2	8
StreamRank	3	1

Table 35-8—TLV types *(continued)*

TLV	TLV type	TLV length
EndStationInterfaces	4	variable
InterfaceID	5	variable
DataFrameSpecification	6	18
IEEE802-MacAddresses	7	12
IEEE802-VlanTag	8	2
IPv4-tuple	9	15
IPv6-tuple	10	39
TrafficSpecification	11	9
TSpecTimeAware	12	12
UserToNetworkRequirements	13	5
InterfaceCapabilities	14	variable
Listener	15	variable
Status	16	variable
StatusInfo	17	3
AccumulatedLatency	18	4
InterfaceConfiguration	19	variable
TimeAwareOffset	20	4
FailedInterfaces	21	variable

The TLV length shall contain the number of octets in the TLV value that follows. The TLV value is specified in subsequent subclauses for each TLV.

The following Backus-Naur Form (BNF) production gives the formal description of the MSRPDU FirstValue structure for the Talker Enhanced attribute:

```
TalkerEnhanced_FirstValue ::= Talker, Status
Talker ::= Type, Length,
           StreamID,
           StreamRank,
           [ EndStationInterfaces, ]
           [ DataFrameSpecification, ]
           TrafficSpecification,
           [ TSpecTimeAware, ]
           [ UserToNetworkRequirements, ]
           [ InterfaceCapabilities, ]
Status ::= Type, Length,
           StatusInfo,
           AccumulatedLatency
           [ InterfaceConfiguration ]
           [ FailedInterfaces ]
```

StreamID ::= Type, Length, *value specified in 35.2.2.10.2*
StreamRank ::= Type, Length, *value specified in 35.2.2.10.3*
EndStationInterfaces ::= Type, Length, *value specified in 35.2.2.10.435.2.2.10.4*
DataFrameSpecification ::= Type, Length, *value specified in 35.2.2.10.535.2.2.10.5*
TrafficSpecification ::= Type, Length, *value specified in 35.2.2.10.6*
TSpecTimeAware ::= Type, Length, *value specified in 35.2.2.10.6*
UserToNetworkRequirements ::= Type, Length, *value specified in 35.2.2.10.7*
InterfaceCapabilities ::= Type, Length, *value specified in 35.2.2.10.835.2.2.10.8*
StatusInfo ::= Type, Length, *value specified in 35.2.2.10.9*
AccumulatedLatency ::= Type, Length, *value specified in 35.2.2.10.10*
InterfaceConfiguration ::= Type, Length, *value specified in 35.2.2.10.11*
FailedInterfaces ::= Type, Length, *value specified in 35.2.2.10.12*
Type BYTE ::= *corresponding TLV Type from Table 35-8*
Length BYTE ::= *corresponding TLV Length (number of octets in TLV Value)*

The following Backus-Naur Form (BNF) production gives the formal description of the MSRPDU FirstValue structure for the Listener Enhanced attribute:

ListenerEnhanced_FirstValue ::= Listener, Status
Listener ::= Type, Length,
StreamID,
[EndStationInterfaces,]
[UserToNetworkRequirements,]
[InterfaceCapabilities]
StatusGroup ::= Type, Length,
StatusInfo,
[InterfaceConfiguration]
[FailedInterfaces]
StreamID ::= Type, Length, *value specified in 35.2.2.10.2*
EndStationInterfaces ::= Type, Length, *value specified in 35.2.2.10.435.2.2.10.4*
UserToNetworkRequirements ::= Type, Length, *value specified in 35.2.2.10.7*
InterfaceCapabilities ::= Type, Length, *value specified in 35.2.2.10.835.2.2.10.8*
StatusInfo ::= Type, Length, *value specified in 35.2.2.10.9*
InterfaceConfiguration ::= Type, Length, *value specified in 35.2.2.10.11*
FailedInterfaces ::= Type, Length, *value specified in 35.2.2.10.12*
Type BYTE ::= *corresponding TLV Type from Table 35-8*
Length BYTE ::= *corresponding TLV Length (number of octets in TLV value)*

35.2.2.10.2 StreamID

The StreamID group is specified in 46.2.3.1.

Figure 35-4 specifies the encoding of the value for the StreamID TLV.

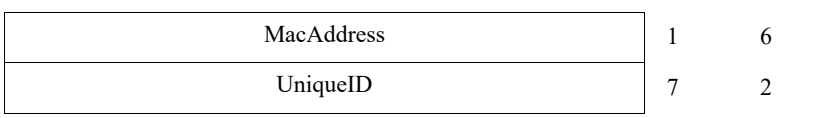

Figure 35-4—Value of StreamID TLV

The MacAddress is encoded in a manner consistent with the MAC address of a frame header.

The semantics of the MacAddress and UniqueID elements of the enhanced StreamID TLV are the same as the semantics of the Mac Address and Unique ID of the original StreamID structure (35.2.2.8.2).

Requirements for incrementing UniqueID are specified in 35.2.2.10.

35.2.2.10.3 StreamRank

The StreamRank group is specified in 46.2.3.2.

Figure 35-5 specifies the encoding of the value for the StreamRank TLV.

	Octet	Length
reserved	1	7 bits
Rank	1	1 bit

Figure 35-5—Value of StreamRank TLV

The semantics of the Rank element of the enhanced StreamRank TLV are the same as the semantics of the Rank of the original PriorityAndRank structure (35.2.2.8.5).

35.2.2.10.4 EndStationInterfaces

The EndStationInterfaces group is specified in 46.2.3.3.

The Value of the EndStationInterfaces TLV shall consist of one or more InterfaceID TLVs. An InterfaceID TLV is provided for each interface used by the Stream's Talker/Listener.

Figure 35-6 specifies the encoding of the value for the InterfaceID TLV.

Group length: 6 + NameLength

	Octet	Length
MacAddress	1	6
InterfaceName	7	NameLength

Figure 35-6—Value of InterfaceID TLV

The MacAddress is encoded in a manner consistent with the MAC address of a frame header.

If the InterfaceName element is not used, its NameLength is zero, and InterfaceName does not exist in the InterfaceID's Value.

EndStationInterfaces does not exist in the original Attribute Types (MSRPv0). EndStationInterfaces enhances MSRP for optional support of CNC.

EndStationInterfaces specifies the identification of physical interfaces (distinct points of attachment) in the end station acting as a Talker/Listener. EndStationInterfaces is used by the CNC to locate the Talker/Listener in the topology. EndStationInterfaces can also be used by end stations with two or more interfaces.

When the externalControl attribute is TRUE (12.32.4.1), EndStationInterfaces should be included in the attribute declared by the Talker or Listener. If the CNC does not have the information needed to identify the Talker/Listener Ports in the topology, the CNC is likely to fail the Stream reservation (35.2.2.10).

When the externalControl attribute is FALSE (12.32.4.1), if EndStationInterfaces contains more than one interface, the Nearest Bridge shall fail the Stream reservation. The fully distributed model does not specify support for multiple interfaces. To fail the Stream reservation, the Nearest Bridge shall change Talker Advertise to Talker Failed and report Failure Code=25 (Table 46-15) and also change Listener Ready or Listener Ready Failed to Listener Asking Failed and report Failure Code=25.

Since the EndStationInterfaces is only used by MRP External Control (12.32.4) for communication with a CNC, the EndStationInterfaces TLV shall be removed from the MSRP FirstValue prior to propagation with MAP (35.2.4).

35.2.2.10.5 DataFrameSpecification

The DataFrameSpecification group is specified in 46.2.3.4.

The TLV for DataFrameSpecification shall consist of Type and Length, followed by a Value that consists of a list of one or more TLVs for each field in the user's frame. The list of TLVs within the DataFrameSpecification value shall be ordered from start of frame to end of header.

Figure 35-7 specifies the encoding of the value for the IEEE802-MacAddresses TLV. The MAC addresses are encoded in a manner consistent with the MAC address of a frame header.

	Octet	Length
DestinationMacAddress	1	6
SourceMacAddress	7	6

Figure 35-7—Value of IEEE802-MacAddresses TLV

Figure 35-8 specifies the encoding of the value for the IEEE802-VlanTag TLV.

	Octet	Length
Priority Code Point	1	3 bits
Reserved	1	1 bit
VLAN ID	1	12 bits

Figure 35-8—Value of IEEE802-VlanTag TLV

Figure 35-9 specifies the encoding of the value for the IPv4-tuple TLV. The IP addresses are encoded in a manner consistent with the IP address of a packet header.

	Octet	Length
SourceIpAddress	1	4
DestinationIpAddress	5	4
Dscp	9	1
Protocol	10	2
SourcePort	12	2
DestinationPort	14	2

Figure 35-9—Value of IPv4-tuple TLV

Figure 35-10 specifies the encoding of the value for the IPv6-tuple TLV. The IP addresses are encoded in a manner consistent with the IP address of a packet header.

	Octet	Length
SourceIpAddress	1	16
DestinationIpAddress	17	16
Dscp	33	1
Protocol	34	2
SourcePort	36	2
DestinationPort	38	2

Figure 35-10—Value of IPv6-tuple TLV

The semantics of the DestinationMacAddress of the enhanced DataFrameSpecification TLV (IEEE802-MacAddresses sub-TLV) are the same as the semantics of the destination_address of the original DataFrameParameters structure (35.2.2.8.3). The semantics of the VlanId of the enhanced DataFrameSpecification TLV (IEEE802-VlanTag sub-TLV) are the same as the semantics of the vlan_identifier of the original DataFrameParameters structure (35.2.2.8.3). The semantics of the PriorityCodePoint element of the enhanced DataFrameSpecification TLV are the same as the semantics of the Data Frame Priority of the original PriorityAndRank structure (35.2.2.8.5).

Requirements for incrementing destination addresses of DataFrameSpecification are specified in 35.2.2.10.

In order to apply TSN behavior to Streams (e.g., reserved bandwidth guarantees), MSRP must be able to distinguish one Stream from another Stream and to distinguish Streams from non-TSN traffic (e.g., best-effort). This requires unique identification of each Stream within the user (i.e., Talker and Listener) as well as the network (i.e., Bridges).

As compared to the original Attribute Types (MSRPv0), the enhancements of the DataFrameSpecification TLV support Stream transformation (46.1.4).

Stream transformation is an optional feature in MSRP Participants. Since Stream Transformation can be supported in either end station or Bridge, one of the following methodologies may be chosen:

a) **No Stream Transformation**

When Stream transformation (46.1.4) is not supported, the user's identification of the Stream must be the same as the network's identification (i.e., destination MAC address and VLAN ID). This is the methodology used for MSRPv0. This methodology requires the following in each end station (Talker/Listener):

1) The destination MAC address of the user’s data frame shall be either multicast (group) or a locally administered unicast (individual). A multicast MAC address is allocated by the user (e.g., a higher layer protocol like IEEE Std 1722). Only one Stream is allowed per destination MAC address. The destination MAC address shall be provided as the DestinationMacAddress element of the IEEE802-MacAddresses sub-TLV of the DataFrameSpecification TLV.

NOTE—When Stream transformation is not supported, MSRP prohibits a DestinationMacAddress element that is universal unicast (i.e., individual MAC address of the Listener). Since the Listener’s universal unicast MAC address is used for non-TSN traffic, it does not uniquely identify the Stream. The VLAN ID alone does not distinguish the Stream’s traffic because Bridges can be configured to egress non-TSN traffic using the same VLAN ID that the Stream is using.

2) The Talker uses its individual MAC address as the source MAC address of the data frame. The source MAC address shall be provided as the SourceMacAddress element of the IEEE802-MacAddresses sub-TLV of the DataFrameSpecification TLV.

3) The data frame shall use a VLAN tag containing a Priority Code Point (PCP) and a VLAN ID, both provided in the IEEE802-VlanTag sub-TLV of the DataFrameSpecification TLV. The Talker uses the value SRclassPriority of the Domain attribute (35.2.2.9.3) for the PriorityCodePoint. If the Talker has knowledge of a specific VLAN ID to use, any valid VlanId can be specified (1 to 4094). If the Talker does not have knowledge of a specific VLAN ID to use, the VlanId uses the value from the SRclassVID of the Domain attribute (35.2.2.9.4).

b) **Stream Transformation in end station**

When Stream transformation (46.1.4) is performed in the end station (Talker/Listener), the end station relies on the network to provide the network's identification of the Stream (i.e., destination MAC address and VLAN ID). The network identification is provided by the CNC through the use of MRP External Control (12.32.4) in the Nearest Bridge (8.6.3). This methodology requires the following in each end station (Talker/Listener):

1) The Talker's DataFrameSpecification TLV shall not be included in the Talker attribute. This indicates to the CNC that Stream transformation is performed in the end station. Since the user's data frame identification is not used in the network, the DataFrameSpecification is not relevant to Bridges.

2) The end station's InterfaceCapabilities TLV (46.2.3.7) shall set VlanTagCapable=true and CB-StreamIdenTypeList containing the Active Destination MAC and VLAN Stream identification type (46.2.3). If the user's data frame uses either IPv4-tuple or IPv6-tuple, the CB-StreamIdenTypeList shall contain the IP Stream identification type. These InterfaceCapabilities specify that the end station supports the IEEE 802.1CB stream identification functions that transform between the user's identification and the corresponding network identification.

3) When the externalControl attribute is FALSE (12.32.4.1), if the Talker's DataFrameSpecification TLV is missing from Talker Advertise, the Nearest Bridge shall fail the Stream reservation. The fully distributed model does not specify support for stream transformation in end station. To fail the Stream reservation, the Nearest Bridge shall change Talker Advertise to Talker Failed and report Failure Code=22 (Table 46-15).

4) The CNC shall allocate a multicast MAC address for the network identification and use MRP External Control to declare an MSRP attribute containing a InterfaceConfiguration sub-TLV (35.2.2.10.11) with configuration values for IEEE802-MacAddresses and IEEE802-VlanTag. These configuration values provide the network identification for the end station's transformation. For an end station acting as Talker, the Nearest Bridge includes InterfaceConfiguration in the declared Listener Ready or Listener Ready Failed. For an end station acting as Listener, the Nearest Bridge includes InterfaceConfiguration in the declared Talker Advertise.

5) When the Stream is withdrawn (i.e., MRP Leave of Talker Advertise), the CNC can free the multicast MAC address for use by subsequent Streams.

c) **Stream Transformation in Bridge**

When Stream transformation (46.1.4) is performed in the Bridge, the end station relies on the CNC to configure the Bridge to perform all transformation. The user identification (i.e.,

DataFrameSpecification) is provided to the CNC through the use of MRP External Control (12.32.4) in the Nearest Bridge (8.6.3). The CNC uses management to configure IEEE 802.1CB features in the Nearest Bridge to transform the Stream between user and network identification. This methodology requires the following:

1) The Talker's DataFrameSpecification TLV provides unique user identification of the Stream, but that user identification does not meet the requirements for no stream transformation [item a) in 35.2.2.10.5]. If the destination address alone does not distinguish the Stream from other Streams and non-TSN traffic, other elements of the DataFrameSpecification shall be unique. For example, if the data frame is untagged with unicast DestinationMacAddress and unicast DestinationIpAddress, a unique DestinationPort can distinguish the Stream.

2) When the externalControl attribute is FALSE (12.32.4.1), since the Talker's DataFrameSpecification TLV does not meet the requirements for no stream transformation [item a) in 35.2.2.10.5], the Nearest Bridge shall fail the Stream reservation. The fully distributed model does not specify support for stream transformation in Bridge. To fail the Stream reservation, the Nearest Bridge shall change Talker Advertise to Talker Failed and report Failure Code=22 (Table 46-15).

3) If the CNC determines that the Nearest Bridge does not support IEEE Std 802.1CB, it shall use MRP External Control to declare an MSRP attribute containing a StatusInfo sub-TLV (35.2.2.10.9) with Failure Code=23 (Table 46-15). For an end station acting as Talker, the Nearest Bridge declares Listener Asking Failed. For an end station acting as Listener, the Nearest Bridge decleares Talker Failed.

4) If the CNC determines that the Nearest Bridge does not support the IEEE 802.1CB stream identification types for the DataFrameSpecification, it shall use MRP External Control to declare an MSRP attribute containing a StatusInfo sub-TLV (35.2.2.10.9) with Failure Code=24 (Table 46-15). For an end station acting as Talker, the Nearest Bridge declares Listener Asking Failed. For an end station acting as Listener, the Nearest Bridge declares Talker Failed.

5) If the CNC successfully configures the IEEE 802.1CB stream identification types for the DataFrameSpecification, it shall use MRP External Control to declare a successful MSRP attribute (assuming no other failures). For an end station acting as Talker, the Nearest Bridge declares Listener Ready or Listener Ready Failed. For an end station acting as Listener, the Nearest Bridge declares Talker Advertise.

6) When the Stream is withdrawn (i.e., MRP Leave of Talker Advertise), the CNC can free the multicast MAC address for use by subsequent Streams. The CNC can also remove IEEE 802.1CB configurations for the withdrawn Stream.

35.2.2.10.6 TrafficSpecification and TSpecTimeAware

The TrafficSpecification group and optional TSpecTimeAware group are specified in 46.2.3.5.

The TrafficSpecification shall consist of a required TrafficSpecification TLV, followed by an optional TSpecTimeAware TLV.

Figure 35-11 specifies the encoding of the value for the TrafficSpecification TLV.

	Octet	Length
Interval	1	4
MaxFramesPerInterval	5	2
MaxFrameSize	7	2
TransmissionSelection	9	1

Figure 35-11—Value of TrafficSpecification TLV

Figure 35-12 specifies the encoding of the value for the TSpecTimeAware TLV. The presence of the optional TSpecTimeAware TLV is handled as specified in 46.2.3.5 for the presence of the TSpecTimeAware group.

	Octet	Length
EarliestTransmitOffset	1	4
LatestTransmitOffset	5	4
Jitter	9	4

Figure 35-12—Value of TSpecTimeAware TLV

The semantics of the MaxFramesPerInterval element of the enhanced TrafficSpecification TLV are the same as the semantics of the MaxIntervalFrames of the original TSpec structure (35.2.2.8.4). The semantics of the MaxFrameSize element of the enhanced TrafficSpecification TLV are the same as the semantics of the MaxFrameSize of the original TSpec structure (35.2.2.8.4).

A Talker that assumes usage of the fully distributed model should use TrafficSpecification as follows:

a) Set the Interval element of the TrafficSpecification TLV to the classMeasurementInterval (34.3) of the stream's SR class (35.2.2.9.2).
b) Set the TransmissionSelection element to the value 1 for use of the credit-based shaper (8.6.8.2).
c) Do not use the TSpecTimeAware TLV.

When the externalControl attribute is FALSE (12.32.4.1), if the Talker's TrafficSpecification TLV does not meet the preceding recommendations a) through c), a Bridge may fail the Stream reservation. Bridges using the fully distributed model are not required to support Talker-specific intervals or shaping beyond those of MSRPv0. To fail the Stream reservation, the Bridge shall change Talker Advertise to Talker Failed and report Failure Code=25 (Table 46-15).

NOTE—In IEEE Std 802.1BA [B13], the AccumulatedLatency computation assumes consistent use of the credit-based shaper at the classMeasurementInterval for the Talker and all Bridges using the SR class. This assumption aligns with the preceding recommendations. The Talker's TrafficSpecification specifies the interval, shaping, and/or scheduling of the Talker only, and TrafficSpecification does not change as it propagates through Bridges. Therefore, when MRP External Control is not in use, in order to support TrafficSpecifications beyond those of IEEE Std 802.1BA, each Bridge must be able to 1) distinguish whether the Port registering Talker Advertise is connected directly to the Talker (i.e., end station) or another Bridge, to determine where TrafficSpecification impacts the AccumulatedLatency, and 2) understand the shaping and scheduling used in each Bridge in the path from Talker to the current Bridge since that also impacts AccumulatedLatency. Specifications for these tasks are not provided in this standard.

35.2.2.10.7 UserToNetworkRequirements

The UserToNetworkRequirements group is specified in 46.2.3.6.

Figure 35-13 specifies the encoding of the value for the UserToNetworkRequirements TLV.

	Octet	Length
NumSeamlessTrees	1	1
MaxLatency	2	4

Figure 35-13—Value of UserToNetworkRequirements TLV

UserToNetworkRequirements does not exist in the original Attribute Types (MSRPv0). UserToNetworkRequirements provides an optional mechanism for the Stream's Talker and/or Listeners to specify requirements to the network.

As specified in the MSRP structure definition (35.2.2.10.1), the UserToNetworkRequirements TLV may be used for either Talker or Listener attributes.

If UserToNetworkRequirements.MaxLatency is greater than zero, when an MSRP Participant detects that the AccumulatedLatency (35.2.2.10.10) of the Stream is greater than UserToNetworkRequirements.MaxLatency, the Stream's status is reported as failed (35.2.4.6).

If UserToNetworkRequirements.MaxLatency is zero (or UserToNetworkRequirements is not provided), when an MSRP Participant detects that the AccumulatedLatency of the Stream is greater than the initial AccumulatedLatency when the Stream was successfully configured, the Stream's status is reported as failed. This technique is compatible with MSRPv0 (35.2.2.8.6).

As specified in 46.2.3.6 and 35.2.4.6, UserToNetworkRequirements.MaxLatency specified by a Talker applies to all Listeners of the stream. UserToNetworkRequirements.MaxLatency specified by a Listener applies to that Listener alone.

Additional specifications for UserToNetworkRequirements.MaxLatency are provided in 35.2.4.4.5.

A Talker that assumes usage of the fully distributed model should use UserToNetworkRequirements as follows:

a) Set the NumSeamlessTrees element to one.

When the externalControl attribute is FALSE (12.32.4.1), if the Talker's UserToNetworkRequirements TLV does not meet the preceding recommendation a), a Bridge may fail the Stream reservation. Bridges using the fully distributed model are not required to support configuration of seamless redundancy. To fail the Stream reservation, the Bridge shall change Talker Advertise to Talker Failed and report Failure Code=25 (Table 46-15).

NOTE—NumSeamlessTrees greater than one is intended for use with MRP External Control (i.e., externalControl TRUE), which passes SRP attributes to a CNC, and the CNC configures Bridges for seamless redundancy (e.g., IEEE Std 802.1CB or IEC 62439-3:2016 [B79]). When MRP External Control is not in use, in order to support NumSeamlessTrees greater than one, each Bridge must be able to either 1) determine that seamless redundancy has already been configured for all Bridges between the Talker and its Listeners or 2) solicit such configuration from an external entity. Specifications for these tasks are not provided in this standard.

35.2.2.10.8 InterfaceCapabilities

The InterfaceCapabilities group is specified in 46.2.3.7.

Figure 35-14 specifies the encoding of the value for the InterfaceCapabilities TLV. NumITL represents the number of elements in the CB-StreamIdenTypeList. NumSTL represents the number of elements in the CB-SequenceTypeList.

TLV Length: 3 + (4 × NumITL) + (4 × NumSTL)	Octet	Length
reserved	1	7 bits
VlanTagCapable	1	1 bit
NumITL	2	1
NumSTL	3	1
CB-StreamIdenTypeList	4	4 × NumITL
CB-SequenceTypeList	4	4 × NumSTL

Figure 35-14—Value of InterfaceCapabilities TLV

InterfaceCapabilities does not exist in the original Attribute Types (MSRPv0). InterfaceCapabilities provides an optional mechanism for stream transformation as specified in 35.2.2.10.5.

The InterfaceCapabilities TLV is used only for negotiation between an end station and the neighboring MRP Participant on the link. The InterfaceCapabilities TLV shall be removed from the MSRP FirstValue prior to propagation with MAP (35.2.4).

35.2.2.10.9 StatusInfo

The StatusInfo group is specified in 46.2.5.1.

Figure 35-15 specifies the encoding of the value for the StatusInfo TLV.

	Octet	Length
TalkerStatus	1	1
ListenerStatus	2	1
FailureCode	3	1

Figure 35-15—Value of StatusInfo TLV

The semantics of the FailureCode element of the enhanced StatusInfo TLV are the same as the semantics of the Failure Code of the original FailureInformation structure (35.2.2.8.7), and both use the same Table 46-15 for values.

The original Attribute Types (MSRPv0) did not provide a FailureCode from a Listener. The enhanced StatusInfo provides a mechanism for a Listener to provide a FailureCode as status to the Talker.

The StatusInfo in the Talker attribute is used to communicate status information to Listeners. The StatusInfo in the Listener attribute is used to communicate status information to the Talker. In SRP, status flows in one direction. Unless stated otherwise in these specifications, elements of StatusInfo are not copied from a Talker attribute to a Listener attribute, or vice versa.

A StatusInfo TLV for Declaration Type (35.2.1.3) of Talker Advertise shall use a TalkerStatus of Ready.

A StatusInfo TLV for Declaration Type of Talker Failed shall use a TalkerStatus of Failed.

A StatusInfo TLV for Declaration Type of Listener Asking Failed shall use a ListenerStatus of Failed.

A StatusInfo TLV for Declaration Type of Listener Ready shall use a ListenerStatus of Ready.

A StatusInfo TLV for Declaration Type of Listener Ready Failed shall use a ListenerStatus of PartialFailed.

For Talker and Listener attributes, the value for the FailureCode element is specified in Table 46-15.

Refer to 35.2.4.4.6 for information on merging nonzero FailureCode values from multiple Listeners.

NOTE—MSRPv1 adds new values for FailureCode (Table 46-15) that did not exist in MSRPv0 (i.e., 20 and higher). MSRPv0 participants that register Talker Failed with these values are expected to handle the failure without complete understanding of the cause.

35.2.2.10.10 AccumulatedLatency

The AccumulatedLatency group is specified in 46.2.5.2.

Figure 35-16 specifies the encoding of the value for the AccumulatedLatency TLV.

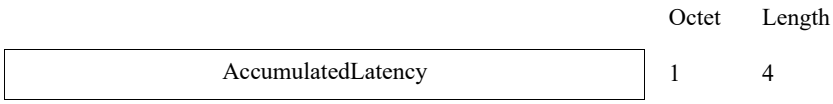

Figure 35-16—Value of AccumulatedLatency TLV

The semantics of the enhanced AccumulatedLatency TLV are the same as the semantics of the original Accumulated Latency structure (35.2.2.8.6).

The original Attribute Types (MSRPv0) failed the Stream when Accumulated Latency increased beyond its initial value. In addition to supporting that mechanism by default, the enhanced AccumulatedLatency (MSRPv1) provides an optional mechanism to fail the Stream when AccumulatedLatency exceeds specified UserToNetworkRequirements for MaxLatency (35.2.2.10.7).

35.2.2.10.11 InterfaceConfiguration

The InterfaceConfiguration group is specified in 46.2.5.3.

The encoding of the value for the InterfaceConfiguration TLV consists of one or more interface configurations. Each interface configuration consists of a single InterfaceID TLV (35.2.2.10.4), followed by a list of configuration values for that interface.

The values in each InterfaceID shall match one of the InterfaceID entries in the Talker/Listener EndStationInterfaces group.

The list of configuration values uses zero or more of the following TLVs:

— IEEE802-MacAddresses TLV (35.2.2.10.5)
— IEEE802-VlanTag TLV (35.2.2.10.5)
— IPv4-tuple TLV (35.2.2.10.5)
— IPv6-tuple TLV (35.2.2.10.5)
— TimeAwareOffset TLV with a value as specified in Figure 35-17

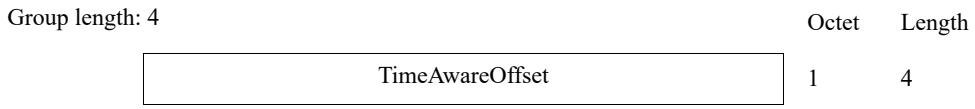

Figure 35-17—Value of TimeAwareOffset TLV

InterfaceConfiguration does not exist in the original Attribute Types (MSRPv0). InterfaceConfiguration provides an optional mechanism for stream transformation as specified in 35.2.2.10.5.

The InterfaceConfiguration TLV is used only for negotiation between an end station and the neighboring MRP Participant on the link. The InterfaceConfiguration TLV shall be removed from the MSRP FirstValue prior to propagation with MAP (35.2.4).

35.2.2.10.12 FailedInterfaces

The FailedInterfaces group is specified in 46.2.5.4.

The FailedInterfaces TLV may be contained within the Talker or Listener attribute. When a failure occurs in network configuration (i.e., nonzero FailureCode in StatusInfo TLV), FailedInterfaces provides a list of one or more physical interfaces (distinct points of attachment) in the failed Bridge or end station. Each entry in the list is sufficient to locate the interface in the physical topology.

The Value of the FailedInterfaces TLV consists of a sequence of zero or more TLV entries, and each TLV entry uses the InterfaceID TLV specified in 35.2.2.10.4.

The semantics of the enhanced FailedInterfaces TLV is similar to the system identifier of the original FailureInformation structure (35.2.2.8.7), but with enhanced ability to identify more than one failed interface.

The FailedInterfaces TLV is associated with the single FailureCode in the StatusInfo TLV, and it is not used to communicate multiple failures.

Elements of FailedInterfaces are not copied from a Talker attribute to a Listener attribute, or vice versa.

35.2.3 Provision and support of Stream registration service

35.2.3.1 Initiating MSRP registration and de-registration

Change the first four paragraphs of 35.2.3.1, and insert the additional paragraphs as shown:

MSRP utilizes the following five declaration types (35.2.1.3) to communicate: Talker Advertise, Talker Failed, Listener Ready, Listener Ready Failed, and Listener Asking Failed.

When a Port's *neighborProtocolVersion* is 0x00, declarations shall use only the original Attribute Types (35.2.2.8). When a Port's *neighborProtocolVersion* is 0x01 or higher, declarations shall use either the original Attributes Types or the enhanced Attributes Types (35.2.2.10).

An end station ~~SR Station~~ behaving as a Talker will send a Talker Advertise declaration to inform the network about the characteristics ~~(35.2.2.8)~~ of the Stream it can provide. Bridges register this declaration, update some of the information contained within the Talker declaration, and forward it out the nonblocked (35.1.3.1) ~~p~~Ports on the Bridge. If *talkerPruning* is enabled and the Bridge has the Stream's destination_address registered on one or more ~~p~~Ports (e.g., using~~via~~ MMRP), it shall only forward the declarations out those ~~p~~Ports. Eventually the Talker declaration will be registered by other end ~~SR~~ stations.

If *talkerPruning* is enabled and the Stream's destination_address is not registered on any ~~p~~Ports, the Talker declaration shall not be forwarded (35.2.4.3.2). When the *talkerPruning* parameter is disabled (false) for the Bridge, each Port provides a *talkerPruningPerPort* parameter to control MAC address pruning for that Port (35.2.4.3.3).

If *talkerVlanPruning* is enabled (35.2.4.3.4), the Bridge will limit the Talker declarations to Ports that have the Stream's VLAN identifier [item b) in 35.2.2.8.3] registered as a member in the VLAN Registration Entries (8.8).

An end station ~~SR Station~~ behaving as a Listener will receive the Talker Advertise declaration and register it. If the Listener is interested in receiving that Stream, it will send a Listener Ready declaration back toward the Talker. The Bridge's MSRP MAP function will use the StreamID (35.2.2.8.2, 35.2.2.10.2) to associate the Listener Ready with the Talker Advertise and forward the Listener Ready declaration only on the ~~p~~Port that registered the Talker Advertise. This is referred to as Listener Pruning since the declarations are not forwarded out any other ~~p~~Ports. The Bridge will also reserve the required bandwidth and configure its queues and the FDB.

35.2.3.1.1 REGISTER_STREAM.request

Change the third paragraph of 35.2.3.1.1 as shown:

On receipt of a REGISTER_STREAM.request the MSRP Participant shall issue a MAD_Join.request service primitive (10.2, 10.3). The attribute_type (10.2) parameter of the request shall carry ~~the value of~~ the appropriate Talker ~~Advertise Vector Attribute Type [item a) in 35.2.2.4] or Talker Failed Vector~~ Attribute Type ~~[item b) in~~ (35.2.2.4~~]~~), depending on the Declaration Type and *neighborProtocolVersion*. The attribute_value (10.2) parameter shall carry the values from the REGISTER_STREAM.request primitive.

35.2.3.1.2 REGISTER_STREAM.indication

Change the second paragraph of 35.2.3.1.2 as shown:

On receipt of a MAD_Join.indication service primitive (10.2, 10.3) with an attribute_type of Talker Advertise, ~~Vector Attribute Type or~~ Talker Failed, or ~~Vector Attribute Type~~ Talker Enhanced (35.2.2.4), the MSRP application shall issue a REGISTER_STREAM.indication to the Listener application entity. The REGISTER_STREAM.indication shall carry the values from the attribute_value parameter.

35.2.3.1.4 DEREGISTER_STREAM.indication

Change the second paragraph of 35.2.3.1.4 as shown:

On receipt of a MAD_Leave.indication service primitive (10.2, 10.3) with an attribute_type of Talker Advertise, ~~Vector Attribute Type or~~ Talker Failed, or ~~Vector Attribute Type~~ Talker Enhanced (35.2.2.4), the MSRP application shall issue a DEREGISTER_STREAM.indication to the Listener application entity.

35.2.3.1.5 REGISTER_ATTACH.request

Change the second paragraph of 35.2.3.1.5 as shown:

On receipt of a REGISTER_ATTACH.request, the MSRP Participant shall issue a MAD_Join.request service primitive (10.2, 10.3). The attribute_type parameter of the request shall carry ~~the value of~~ the appropriate Listener ~~Vector~~ Attribute Type ~~[item c) in~~ (35.2.2.4~~]~~), depending on *neighborProtocolVersion*. The attribute_value shall contain the StreamID and the Declaration Type.

35.2.3.1.6 REGISTER_ATTACH.indication

Change the second paragraph of 35.2.3.1.6 as shown:

On receipt of a MAD_Join.indication service primitive (10.2, 10.3) with an attribute_type of Listener ~~Vector Attribute Type~~ (35.2.2.4), the MSRP application shall issue a REGISTER_ATTACH.indication to the Talker application entity. The REGISTER_ATTACH.indication shall carry the values from the attribute_value parameter.

35.2.3.1.7 DEREGISTER_ATTACH.request

Change the second paragraph of 35.2.3.1.7 as shown:

On receipt of a REGISTER_STREAM.request, the MSRP Participant shall issue a MAD_Leave.request service primitive (10.2, 10.3) with the attribute_type set to the appropriate Listener ~~Vector~~ Attribute Type (35.2.2.4). The attribute_value parameter shall carry the StreamID and the Declaration Type currently associated with the StreamID.

35.2.3.1.8 DEREGISTER_ATTACH.indication

Change the second paragraph of 35.2.3.1.8 as shown:

On receipt of a MAD_Leave.indication service primitive (10.2, 10.3) with an attribute_type of Listener ~~Vector Attribute Type~~ (35.2.2.4), the MSRP application shall issue a DEREGISTER_ATTACH.indication to the Talker application entity. The DEREGISTER_ATTACH.indication shall contain the StreamID.

35.2.4 MSRP Attribute Propagation

Change the first four paragraphs (and lists) of 35.2.4 as shown:

There is no MSRP MAP function for Domain attributes. MSRP simply declares the characteristics of the SR classes that are supported on the Bridge Port regardless of what has been learned from Domain registrations on other Bridge Ports. Registration of the neighbor's Domain attribute determines the value of the *SRPDomainBoundaryPort* and *neighborProtocolVersion* variables (35.2.1.4).

This subclause describes the following:

a) Rules for combining and propagating Listener attributes toward the associated Talker.
b) How MSRP adjusts the Talker and Listener attributes before propagating them.
c) How MSRP translates Attribute Types when propagating from a ProtocolVersion (35.2.2.3) on a registering Port to a different ProtocolVersion on declaring Ports.

Unless stated otherwise, Talker and Listener attributes are propagated as described in 10.3.

In principle, the MAP performs MSRP Attribute Propagation when any of the following conditions occur:

d) ~~c)~~ A MAD_Join.indication adds a new attribute to MAD (with the new parameter, 10.2, set to TRUE).
e) ~~d)~~ A MAD_Leave.indication is issued by the MAD.
f) ~~e)~~ An internal application declaration or withdrawal is made in a station.
g) ~~f)~~ When the bandwidth of the underlying media changes (see bandwidthAvailabilityChanged notification in 34.3.2).
h) ~~g)~~ A ~~p~~Port becomes an SRP domain core port (3.258).
i) ~~h)~~ If *talkerPruning* (~~35.2.4.3.1~~35.2.4.3.2) or *talkerPruningPerPort* (35.2.4.3.3) is enabled and there is a change in the MAC Address Registration Entries (8.8.4) of ~~MMRP (10.9) attributes registered on~~ a ~~p~~Port.
j) If *talkerVlanPruning* (35.2.4.3.4) is enabled and there is a change in the VLAN Registration Entries on a Port (8.8).
k) A Port changes its *neighborProtocolVersion* parameter (35.2.1.4(j).

35.2.4.1 Stream importance

Change 35.2.4.1 as shown:

MSRP utilizes the stream Rank [item b) in 35.2.2.8.5, 35.2.2.10.3] to decide which stream is more important than another.

In the case where two streams have the same Rank, the *streamAge* will be compared. If the Ranks are identical and the *streamAge*s are identical, the StreamIDs (35.2.2.8.2, 35.2.2.10.2) will be compared, and the numerically lower StreamID is considered to be more important.

35.2.4.2 Stream bandwidth calculations

Change the last paragraph of 35.2.4.2 as shown:

Streams that are in the Listener Ready or Listener Ready Failed state reduce the amount of bandwidth available to other Streams. Streams that have no Listeners, or the Listeners are in the Asking Failed state, do not reduce available bandwidth for other Streams since the Stream data will not be flowing through this outbound ~~p~~Port. These details are considered when calculating how many Streams can flow through a particular ~~p~~Port. The total amount of bandwidth available to a particular traffic class on a ~~p~~Port is represented by *deltaBandwidth(N)* [item b~~1~~) in 34.3]. Subclauses 34.3.1 and 34.3.2 describe~~s~~ the relationship of available bandwidth between traffic classes.

35.2.4.3 Talker attribute propagation

Insert the following paragraph at the end of the introductory text of 35.2.4.3:

If the outbound Port is an SRP domain boundary port for a given SR class, then for that SR class, the MAP function shall operate such that any Talker Advertise declaration for that SR class that would otherwise propagate through that Port is converted to a Talker Failed declaration with a failure code of 8 (Table 46-15), meaning “Egress Port is not AVB-capable.” The consequence of this behavior is that stream reservations can be established only in circumstances where the Talker and the Listener(s) for the stream are located within the same SRP domain.

Insert the following subclause (35.2.4.3.1, including Table 35-12) after the introductory text of 35.2.4.3, and renumber the subsequent tables in Clause 35 accordingly [see subsequent instructions for changes to the "Talker Pruning" subclause (now 35.2.4.3.2)]:

35.2.4.3.1 ProtocolVersion Translation

Ports with *neighborProtocolVersion* [item j) in 35.2.1.4] of 0x00 shall use only the original Attribute Types (35.2.2.4) of Talker Advertise and Talker Failed. Ports with *neighborProtocolVersion* of 0x01 or higher shall use either the original Attribute Types or the enhanced Attribute Type of Talker Enhanced.

In order to propagate an enhanced Attribute Type (e.g., registered from a Port with *neighborProtocolVersion* of 0x01) to an original Attribute Type (e.g., declared to a Port with *neighborProtocolVersion* of 0x00), fields of the attribute's FirstValue shall be translated as specified in Table 35-12.

NOTE—Original Attribute Types always propagate without translation, even if *neighborProtocolVersion* is 0x01 or higher.

Table 35-12—Translation of Talker attributes

Enhanced field	Original field	Enhanced to original translation
StreamID, MacAddress	StreamID, MAC Address	No change: Copy the field's contents from enhanced to original without change.
StreamID, UniqueID	StreamID, Unique ID	No change: Copy the field's contents from enhanced to original without change.
StreamRank, Rank	PriorityAndRank, Rank	No change: Copy the field's contents from enhanced to original without change.
EndStationInterfaces	—	Not propagated: This TLV is not propagated through MAP, so translation is not applicable.
DataFrameSpecification, IEEE802-MacAddresses, DestinationMacAddress	DataFrameParameters, destination_address	No change: Copy the field's contents from enhanced to original without change.
DataFrameSpecification, IEEE802-MacAddresses, SourceMacAddress	—	Lost: The declaration type does not change (i.e., fail), because MSRPv0 does not use the source MAC address for Stream identification.
DataFrameSpecification, IEEE802-VlanTag, PriorityCodePoint	PriorityAndRank, Data Frame Priority	No change: Copy the field's contents from enhanced to original without change.
DataFrameSpecification, IEEE802-VlanTag, VlanId	DataFrameParameters, vlan_identifier	No change: Copy the field's contents from enhanced to original without change.
TrafficSpecification, Interval	—	Lost: If this field does not match the default classMeasurementInterval (34.3) of the stream's SR class, a Talker Advertise declaration is converted to a Talker Failed declaration with a failure code of 20.
TrafficSpecification, MaxFramesPerInterval	TSpec, MaxIntervalFrames	No change: Copy the field's contents from enhanced to original without change.
TrafficSpecification, MaxFrameSize	TSpec, MaxFrameSize	No change: Copy the field's contents from enhanced to original without change.

Table 35-12—Translation of Talker attributes *(continued)*

Enhanced field	Original field	Enhanced to original translation
TrafficSpecification, TransmissionSelection	—	Lost: If this field is not one, a Talker Advertise declaration is converted to a Talker Failed declaration with a failure code of 20.
TrafficSpecification, TSpecTimeAware TLV	—	Lost: If this TLV is present, a Talker Advertise declaration is converted to a Talker Failed declaration with a failure code of 20.
UserToNetworkRequirements, NumSeamlessTrees	—	Lost: If this field is not one, a Talker Advertise declaration is converted to a Talker Failed declaration with a failure code of 20.
UserToNetworkRequirements, MaxLatency	—	Lost: If this field is not zero, a Talker Advertise declaration is converted to a Talker Failed declaration with a failure code of 20.
InterfaceCapabilities	—	Not propagated: This TLV is not propagated through MAP, so translation is not applicable.
StatusInfo, TalkerStatus	—	No change: This field determines the original Attribute Type to declare (Talker Advertise or Talker Failed).
StatusInfo, ListenerStatus	—	No change: This field is set according to the ListenerStatus that the Bridge last propagated in the Listener attribute of the Stream. If no Listener attribute has been propagated, this field is set to zero (None).
StatusInfo, FailureCode	FailureInformation, Failure Code	For Talker Failed, this field has no change (i.e., copy from enhanced to original). For Talker Advertise, this field is lost, but the declaration type does not change (i.e., fail).
AccumulatedLatency	AccumulatedLatency, AL	No change: Copy the field's contents from enhanced to original without change.
InterfaceConfiguration	—	Not propagated: This TLV is not propagated through MAP, so translation is not applicable.
FailedInterfaces, MacAddress	FailureInformation, system identifier	No change: This field applies to Talker Failed only. The most significant 16 bits of system identifier are set to zero, and the MacAddress is copied to the least significant 48 bits of system identifier.
FailedInterfaces, InterfaceName	FailureInformation	Lost: This field applies to Talker Failed only.

Change the subclause number and text of now 35.2.4.3.2 as shown:

35.2.4.3.2 ~~35.2.4.3.1~~ Talker Pruning

The *talkerPruning* parameter (35.2.1.4) is disabled by ~~By~~ default, and Talker declarations are sent out all nonblocked ~~p~~Ports. If *talkerPruning* is enabled and the destination_address [item a) in 35.2.2.8.3] of the Stream is found in the MAC Address Registration Entries (8.8.4) for the ~~p~~Port, the declaration shall be forwarded. If the destination_address is not found, the declaration shall be blocked, and no Talker declaration of any type shall be forwarded.

Insert the following subclauses (35.2.4.3.3 and 35.2.4.3.4) after 35.2.4.3.2:

35.2.4.3.3 Talker Pruning Per Port

When the *talkerPruning* parameter is disabled (false) for the Bridge, each Port may provide a *talkerPruningPerPort* parameter (35.2.1.4) to control MAC address pruning for that Port.

When the *talkerPruningPerPort* feature is provided, all *talkerPruningPerPort* parameters are disabled by default.

When *talkerPruning* is enabled (true), the *talkerPruningPerPort* parameters are ignored.

Each *talkerPruningPerPort* parameter controls MAC address pruning for that Port when *talkerPruning* is disabled (false). If *talkerPruningPerPort* is disabled for the Port, Talker declarations are forwarded on that Port regardless of the destination_address [item a) in 35.2.2.8.3] of the Stream. If *talkerPruningPerPort* is enabled for the Port and the destination_address of the Stream is found in the MAC Address Registration Entries (8.8.4) for the Port, the declaration shall be forwarded. If *talkerPruningPerPort* is enabled for the Port and the destination_address of the Stream is not found in the MAC Address Registration Entries for the Port, the declaration shall be blocked, and no Talker declaration of any type shall be forwarded.

35.2.4.3.4 Talker VLAN Pruning

The *talkerVlanPruning* parameter (35.2.1.4) may be provided on the Bridge in order to limit the Talker declarations to Ports that have the Stream's vlan_identifier [item b) in 35.2.2.8.3] registered as a member in the VLAN Registration Entries (8.8).

When the *talkerVlanPruning* feature is provided, it is disabled (false) by default.

When *talkerVlanPruning* is disabled (false), the MAP context for MSRP propagates Talker declarations according to the VLAN spanning tree (35.1.3.1), regardless of the registration of the Stream's vlan_identifier in the Filtering Database (8.8.10). This allows potential Listeners to receive Talker declarations prior to registering the VLAN with MVRP.

When *talkerVlanPruning* is enabled (true), in addition to propagating along the VLAN spanning tree, the MAP context for MSRP filters propagation according to the registration of the Stream's vlan_identifier in the Filtering Database (8.8.10). If *talkerVlanPruning* is enabled and registration of the Stream's vlan_identifier on a Port is not member (Registration Forbidden, Not Registered, or No Dynamic Registration Entry Present), MSRP shall not propagate the declaration on that Port. This propagation is effectively the same as propagation for the Stream's data frames. Potential Listeners can join the VLAN dynamically with MVRP in order to receive Talker declarations for that vlan_identifier (35.1.2.1).

The *talkerVlanPruning* parameter filters by VLAN, and it operates independently from the parameters that filter by MAC address (*talkerPruning* and *talkerPruningPerPort*).

35.2.4.4 Listener attribute propagation

Insert the following subclause (35.2.4.4.4, 35.2.4.4.5, and 35.2.4.4.6, including Table 35-17) after 35.2.4.4.3:

35.2.4.4.4 ProtocolVersion Translation

Ports with *neighborProtocolVersion* [item j) in 35.2.1.4] of 0x00 shall use only the original Attribute Type (35.2.2.4) of Listener. Ports with *neighborProtocolVersion* of 0x01 or higher shall use either the original Attribute Type or the enhanced Attribute Type of Listener Enhanced.

In order to propagate an enhanced Attribute Type (e.g., registered from a Port with *neighborProtocolVersion* of 0x01) to an original Attribute Type (e.g., declared to a Port with *neighborProtocolVersion* of 0x00), fields of the attribute's FirstValue shall be translated as specified in Table 35-17.

NOTE—Original Attribute Types always propagate without translation, even if neighborProtocolVersion is 0x01 or higher.

Table 35-17—Translation of Listener attributes

Enhanced field	Original field	Enhanced to original translation
StreamID, MacAddress	StreamID, MAC Address	No change: This field determines the original Attribute Type to declare (Talker Advertise or Talker Failed).
StreamID, UniqueID	StreamID, Unique ID	No change: This field determines the original Attribute Type to declare (Talker Advertise or Talker Failed).
EndStationInterfaces	—	Not propagated: This TLV is not propagated through MAP, so translation is not applicable.
UserToNetworkRequirements, NumSeamlessTrees	—	Lost: If this field is not one, a Listener Ready or Listener Ready Failed declaration is converted to a Listener Asking Failed declaration.
UserToNetworkRequirements, MaxLatency	—	Lost: If this field is not zero, a Listener Ready or Listener Ready Failed declaration is converted to a Listener Asking Failed declaration.
InterfaceCapabilities	—	Not propagated: This TLV is not propagated through MAP, so translation is not applicable.
StatusInfo, TalkerStatus	—	No change: This field is set according to the TalkerStatus that the Bridge last propagated in the Talker attribute of the Stream. If no Talker attribute has been propagated, this field is set to zero (None).
StatusInfo, ListenerStatus	—	No change: This field determines the original FourPackedType (35.2.2.10) to declare.
StatusInfo, FailureCode	—	Lost: The declaration type does not change (i.e., fail).
InterfaceConfiguration	—	Not propagated: This TLV is not propagated through MAP, so translation is not applicable.

35.2.4.4.5 Merge MaxLatency

When more than one Listener provides the UserToNetworkRequirements TLV, multiple values of MaxLatency shall be merged by using the largest MaxLatency value in that set of multiple values. This ensures that Bridges closer to the Talker support the most flexible requirement for any Listener. The requirements of an individual Listener continue to be met in Bridges closer to that Listener.

35.2.4.4.6 Merge Listener Failures

As specified in 35.2.4.4.3, it is possible for a Bridge to merge failed Declaration Types for Listeners (e.g., Ready Failed with Asking Failed). Each failed Declaration Type uses a nonzero FailureCode (35.2.2.10.9) and a FailedInterfaces TLV associated with that FailureCode.

When merging failures, the FailedInterfaces TLV shall always be transferred with its corresponding StatusInfo TLV since the FailureCode is associated with the FailedInterfaces.

When merging a Ready Failed declaration with an Asking Failed declaration, the StatusInfo and FailedInterfaces shall be transferred from the Asking Failed declaration. When merging failed declarations with the same Declaration Type, the StatusInfo and FailedInterfaces shall be transferred from the declaration with the Failure Code of the lowest numeric value. When merging failed declarations with the same Declaration Type and Failure Code, the decision of which failure to transfer is left to the Bridge's implementation.

Insert the following subclause (35.2.4.6) after 35.2.4.5:

35.2.4.6 MaxLatency Comparison

The comparison for UserToNetworkRequirements.MaxLatency greater than zero determines whether to fail the Stream due to AccumulatedLatency (35.2.2.8.6, 35.2.2.10.10) that exceeds the requirement specified by Talkers and/or Listeners. The comparison is performed separately for the Talker and each Listener.

For the Talker, the UserToNetworkRequirements.MaxLatency from the Talker is compared to AccumulatedLatency for all Listeners of the Stream. For example, if the Bridge forwards the Stream to three Ports (three Listeners registered), the Talker's UserToNetworkRequirements.MaxLatency is compared to three values of AccumulatedLatency. For any of the comparisons, if any Listener's AccumulatedLatency is greater than the Talker's UserToNetworkRequirements.MaxLatency, the Talker fails the MaxLatency comparison.

If the Talker fails the MaxLatency comparison and the Declaration Type of the Talker is Advertise, the Bridge shall change Talker Advertise to Talker Failed and report Failure Code=21 (Table 46-15).

For the Listener, the UserToNetworkRequirements.MaxLatency from the Listener is compared to AccumulatedLatency for the corresponding Listener. If the Listener's AccumulatedLatency is greater than the Listener's UserToNetworkRequirements.MaxLatency, the Listener fails the MaxLatency comparison.

If the Listener fails the MaxLatency comparison, the Bridge shall change Listener Ready to Listener Asking Failed for the failed Listener and report Failure Code=21. The Listener Asking Failed is merged with declarations of other Listeners to determine the resultant Declaration Type to the Talker (35.2.4.4.3).

In the fully distributed model (46.1.3.1), the comparison for the Talker's UserToNetworkRequirements.MaxLatency is performed in each Bridge in the path from the Talker to each Listener, in the same direction as AccumulatedLatency. In the fully distributed model, the comparison for the Listener's UserToNetworkRequirements.MaxLatency is performed in each Bridge in the path from the Listener to each Talker, using the AccumulatedLatency from the Talker's attribute.

In the centralized network/distributed user model (46.1.3.2), the comparison for the Talker's and Listener's UserToNetworkRequirements.MaxLatency is performed in the CNC, using the capabilities of each Bridge as learned through remote management (e.g., Bridge Delay of 12.32.1). When the CNC detects failure of the MaxLatency comparison, Failure Code 21 is declared to the Talker and Listener as specified in item a) in 35.2.2.10.

35.2.6 Encoding

Change the first sentence of the first paragraph of 35.2.6 as shown:

If an MSRP message is received from a Port with an event value ~~(35.2.6)~~ specifying the JoinIn or JoinMt message, and if the StreamID (35.2.2.8.2, 35.2.2.10.2) and Direction (35.2.1.2) all match those of an attribute already registered on that Port, and if the Attribute Type (35.2.2.4) or FourPackedEvent (35.2.2.7.2) has changed, then the Bridge should behave as though an **rLv!** event (with immediate leavetimer expiration in the Registrar state table) was generated for the MAD in the Received MSRP Attribute Declarations before the **rJoinIn!** or **rJoinMt!** event for the attribute in the received message is processed.

35.2.7 Attribute value support requirements

Change 35.2.7 as shown:

Implementations of MSRP shall maintain state information for all attribute values that support the Stream registrations (35.2.2.8, 35.2.2.10).

Implementations of MSRP shall be capable of supporting any attribute value in the range of possible values that can be registered using Stream registrations ~~(35.2.2.8)~~; however, the maximum number of attribute values for which the implementation is able to maintain current state information is an implementation decision, and may be different for Talker attributes and Listener attributes. The number of values that the implementation can support shall be stated in the PICS.

Insert the following text (Clause 46) after Clause 45:

46. Time-Sensitive Networking (TSN) configuration

Time-Sensitive Networking (TSN) is a collection of features in IEEE 802.1 standards that provide the following:

— Time synchronization among Bridges and end stations
— Significant reduction in frame loss due to faults in network equipment
— Significant reduction in, or the elimination of, frame loss due to egress Port congestion
— Bounded latency

This clause provides specifications for the configuration of TSN features in a network. The configuration process begins when Talkers and Listeners pass their requirements to the network and proceeds with the configuration of TSN features in Bridges along a tree from each Talker to its Listener(s).

46.1 Overview of TSN configuration

46.1.1 User/Network Interface (UNI)

TSN configuration uses the concept of a Stream that is transmitted by a Talker to one or more Listeners. The Talkers and Listeners are located within end stations.

This clause specifies configuration information that is exchanged over a User/Network Interface (UNI). The user side of the interface represents Talkers and Listeners. The network side of the interface represents the Bridges that transfer frames of the Stream from each Talker to its Listeners. Each user specifies requirements for its data, but without detailed knowledge of the network. The network obtains requirements from users, analyzes the topology and TSN capabilities of Bridges, and configures the Bridges to meet user requirements. The network returns the success or failure of each Stream's configuration to the user.

46.1.2 Modeling of user/network configuration information

A variety of protocols can be used for the exchange of configuration information over the TSN UNI (e.g., signaling protocols, remote network management protocols). These protocols can exchange the configuration information as text or as binary fields. To enable flexible integration of TSN configuration into a variety of protocols, 46.2 specifies the TSN user/network configuration information in a manner that is independent of schema, encoding, or protocol.

Specific TSN-capable products list the user/network protocol that is supported as part of their conformance [e.g., 5.18.3, item c) in 5.29]. Each user/network protocol will specify a specific schema and/or encoding for the configuration information in 46.2. Examples of these protocols are described for each of the TSN configuration models in 46.1.3.

46.1.3 TSN configuration models

This subclause describes three models for TSN user/network configuration. These models provide an architectural context for subsequent specifications. Each model specification shows the logical flow of user/network configuration information between various entities in the network.

46.1.3.1 Fully distributed model

In the fully distributed model, the end stations that contain users of Streams (i.e., Talkers and Listeners) communicate the user requirements directly over the TSN user/network protocol. The network is configured in a fully distributed manner, without a centralized network configuration entity. The distributed network configuration is performed using a protocol that propagates TSN user/network configuration information along the active topology for the Stream (i.e., Bridges in the tree from Talker to Listeners).

As user requirements propagate through each Bridge, management of the Bridge's resources is effectively performed locally. This local management is limited to the information that the Bridge has knowledge of and does not necessarily include knowledge of the entire network.

Figure 46-1 provides a graphical representation of the fully distributed model.

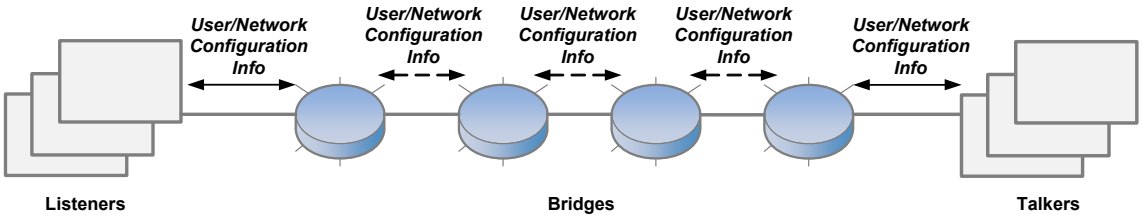

Figure 46-1—Fully distributed model

In the figure, the solid arrows represent the protocol that is used as the UNI for exchange of configuration information between Talkers/Listeners (users) and Bridges (network). This configuration information is specified in 46.2.

In the figure, the dashed arrows represent the protocol that propagates configuration information within the network. This protocol carries the TSN user/network configuration information (46.2) as well as additional information that is specific to network configuration.

The following TSN features can be configured by Bridges using this model:

a) Credit-based shaper algorithm (8.6.8.2) and its configuration (Clause 34)

The Stream Reservation Protocol (SRP) of Clause 35 can be used as the UNI and to propagate configuration information throughout the network of Bridges. SRP exchanges configuration information as binary fields using the Type-Length-Value (TLV) technique. Using this technique, the protocol's top-level message contains a list of one or more TLVs. Each TLV consists of a Type field that specifies what the Value field contains, a Length field that specifies the number of octets in the Value field, and the Value field. In SRP specifications, each TLV Type identifies one of the groups specified in 46.2, and the TLV Value contains a binary representation of the elements in that group.

46.1.3.2 Centralized network/distributed user model

Some TSN use cases are computationally complex. For example, for scheduled traffic (8.6.8.4), computation of the gate control list of each Port can take significant time. For such use cases, it is helpful to centralize the computation in a single entity (Bridge or end station), rather than perform the computation in all Bridges.

Some TSN use cases can benefit from a complete knowledge of all Streams in the network. For example, if the bandwidth for multiple Streams is greater than the available bandwidth along the shortest path between Talkers and Listeners, it is helpful to forward a subset of those Streams along a path other than the shortest. For these use cases, a centralized entity can gather information for the entire network in order to find the best configuration.

The centralized network/distributed user model is similar to the fully distributed model in that end stations communicate their Talker/Listener requirements directly over the TSN UNI. In contrast, in the centralized network/distributed user model, the configuration information is directed to/from a Centralized Network Configuration (CNC) entity. All configuration of Bridges for TSN Streams is performed by this CNC using a remote network management protocol.

The CNC has a complete view of the physical topology of the network as well as the capabilities of each Bridge. This enables the CNC to centralize complex computations. The CNC can exist in either an end station or a Bridge.

The CNC knows the address of all Bridges at the edge of the network (those with an end station connected). The CNC configures those edge Bridges to act as a proxy, transferring Talker/Listener information directly between the edge Bridge and the CNC, rather than propagate the information to the interior of the network.

Figure 46-2 provides a graphical representation of the centralized network/distributed user model.

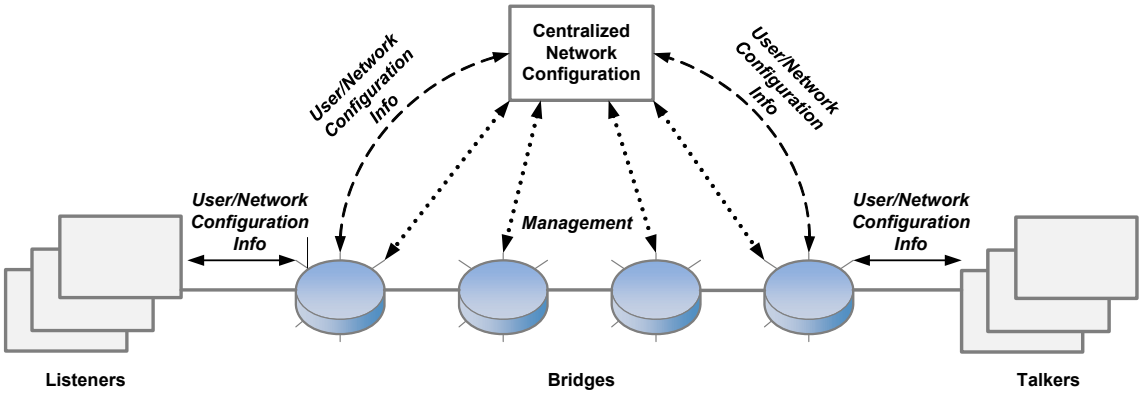

Figure 46-2—Centralized network/distributed user model

In the figure, the solid arrows represent the protocol that is used as the UNI for exchange of configuration information between Talkers/Listeners (users) and Bridges (network). This configuration information is specified in 46.2.

In the figure, the dashed arrows represent the protocol that transfers configuration information between edge Bridges and the CNC. This configuration information is specified in 46.2.

In the figure, dotted arrows represent the remote network management protocol. The CNC acts as the management client, and each Bridge acts as the management server. The CNC uses remote management to discover physical topology, retrieve Bridge capabilities, and configure TSN features in each Bridge. Talkers and Listeners are not required to participate in this remote network management protocol. The information carried by the remote network management protocol is specified in Clause 12.

NOTE 1—If the Talker/Listener protocol of the fully distributed model is selected to be the same as the Talker/Listener protocol of the centralized network/distributed user model, end stations can support both models without explicit knowledge of how the network is configured.

The following TSN features can be configured by the CNC using this model:

a) Credit-based shaper algorithm (8.6.8.2) and its configuration (Clause 34)
b) Frame preemption (6.7.2)
c) Scheduled traffic (8.6.8.4, 8.6.9)
d) Frame Replication and Elimination for Reliability (IEEE Std 802.1CB)
e) Per-stream filtering and policing (8.6.5.1)
f) Cyclic queuing and forwarding (Annex T)

SRP (Clause 35) can be used as the UNI (solid arrows of Figure 46-2). SRP's MRP External Control (12.32.4) feature can be used to exchange configuration information with the CNC component (dashed arrows of Figure 46-2). SRP exchanges configuration information using the TLV technique to reference elements in 46.2 (see 46.1.3.1). Examples of a remote network management protocol (dotted arrows of Figure 46-2) include Simple Network Management Protocol (SNMP), NETCONF (IETF RFC 6241 [B82]), and RESTCONF (IETF RFC 8040 [B85]).

NOTE 2—NETCONF and RESTCONF specify a startup datastore: nonvolatile configuration that is applied when the Bridge powers on. The startup datastore feature enables a TSN CNC to configure Bridges and then remove itself from the network. SNMP does not specify a startup datastore feature. If SNMP is used by a TSN CNC, this can be mitigated by a) using a proprietary (Bridge-specific) startup datastore feature or b) ensuring that the TSN CNC is always active in the network in order to reconfigure Bridges that cycle power.

46.1.3.3 Fully centralized model

Many TSN use cases require significant user configuration in the end stations that act as Talkers and Listeners. For example, in many automotive and industrial control applications, the timing of physical inputs and outputs (I/Os) is determined by the physical environment under control, and the timing requirements for TSN Streams are derived from that I/O timing. In some use cases, these I/O timing requirements can be computationally complex and involve detailed knowledge of the application software/hardware within each end station.

In order to accommodate this sort of TSN use case, the fully centralized model enables a Centralized User Configuration (CUC) entity to discover end stations, retrieve end station capabilities and user requirements, and configure TSN features in end stations. The protocols that the CUC uses for this purpose are specific to the user application and outside the scope of this standard.

From a network perspective, the primary difference between the fully centralized model and the centralized network/distributed user model is that all user requirements are exchanged between the CNC and CUC. Therefore, the TSN UNI exists between the CNC and CUC.

Figure 46-3 provides a graphical representation of the fully centralized model.

In the figure, the solid arrows represent the protocol that is used as the UNI for exchange of configuration information between the CUC and the CNC. This configuration information is specified in 46.2.

In the figure, the dotted arrows represent the remote network management protocol. The CNC acts as the management client, and each Bridge acts as the management server. The CNC uses remote management to discover physical topology, retrieve Bridge capabilities, and configure TSN features in each Bridge. Talkers and Listeners are not required to participate in this remote network management protocol. The information carried by the remote network management protocol is specified in Clause 12.

In this fully centralized model, a protocol is used between the CUC and end stations (Talkers and Listeners) to retrieve end station capabilities and requirements and to configure the end stations. Since that protocol is user-to-user, its configuration information is considered to be outside the scope of this standard, and it is not shown in Figure 46-3.

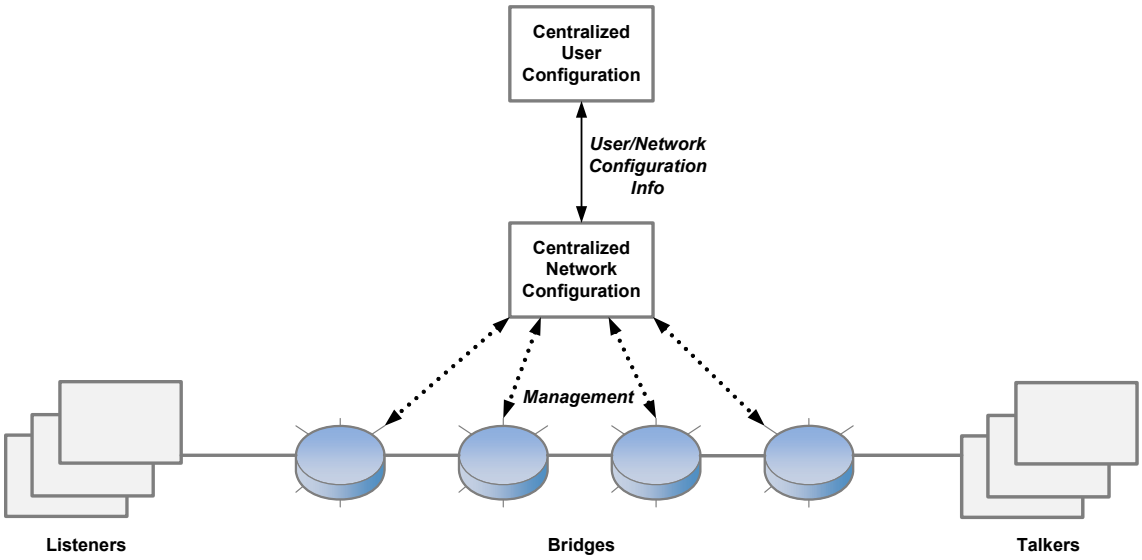

Figure 46-3—Fully centralized model

The following TSN features can be configured by the CNC using this model:

a) Credit-based shaper algorithm (8.6.8.2) and its configuration (Clause 34)
b) Frame preemption (6.7.2)
c) Scheduled traffic (8.6.8.4, 8.6.9)
d) Frame Replication and Elimination for Reliability (IEEE Std 802.1CB)
e) Per-stream filtering and policing (8.6.5.1)
f) Cyclic queuing and forwarding (Annex T)

YANG (IETF RFC 7950) is a data modeling language used to model configuration data and state data for remote network management protocols.[3] The remote network management protocol uses a specific encoding such as XML or JSON. For a particular feature, a YANG module specifies the organization and rules for the feature's management data, and a mapping from YANG to the specific encoding enables the data to be understood correctly by both client (e.g., network manager) and server (e.g., Bridge). Technically speaking, the TSN user/network configuration is not network management, in that information is exchanged between user and network, and not between a network manager and the network's Bridges (Clause 12). Nevertheless, the concepts are sufficiently similar that YANG is useful for modeling the configuration and state data for the TSN user/network configuration information.

In order to support the use of YANG-based protocols for the fully centralized model, 46.3 specifies a YANG module. The YANG module specifies a YANG typedef/grouping for each group of information in 46.2.

NOTE—At the time that this clause was developed, specific protocol implementations for the fully centralized model were a work in progress. One protocol explored for the UNI between CUC and CNC is RESTCONF (IETF RFC 8040 [B85]). A complete YANG module for the TSN UNI can be specified in a document other than IEEE Std 802.1Q. In order to conform with this clause, the complete TSN UNI YANG module imports the YANG module of 46.3 for use within its containers and lists. The complete TSN UNI YANG module will presumably specify features outside the scope of this clause, such as operations to control the deployment of Stream configuration to the network. The JSON encoding can be used with RESTCONF. Although the TSN UNI is technically not network management, use of RESTCONF provides a simple and effective application programming interface (API) for TSN configuration.

For an informative example workflow using the fully centralized model, refer to U.2.

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

46.1.4 Stream transformation

TSN configuration uses the concept of a Stream of data that is transmitted by a Talker to one or more Listeners. The Talkers and Listeners are end stations.

In order to apply TSN behavior to Streams (e.g., reserved bandwidth guarantees), the network must be able to distinguish one Stream from another Stream and distinguish Streams from non-TSN traffic (e.g., best-effort). Therefore, each frame of the Stream must contain fields in its header that uniquely identify the Stream.

The goal of TSN configuration is to allow Talkers and Listeners to use their existing transport layer and application layer protocols for data, rather than requiring a TSN-specific frame format. TSN achieves this goal by identifying each Stream using fields from well-established frame formats such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and IEEE 802.1 (i.e., MAC addresses and VLAN identifier).

As the frames of each Stream cross the user/network boundary, the identification of the Stream in its frames can be different between the network and the user. For example, the user can use UDP without an awareness of VLAN IDs, but the network can require a specific VLAN ID in order to apply TSN features. In order to support this sort of difference in frame format, the TSN user/network configuration information (46.2) provides features to enable transformation of the Stream’s identification at the user/network boundary. The user identification translates to/from the network identification at the boundary, either within the end station or a nearest Bridge. This transformation has the benefit of allowing the user's identification to match its higher layer application protocol and the network's identification to match the bridging technology.

Stream transformation can be accomplished using the functions specified in IEEE Std 802.1CB. The functions of IEEE Std 802.1CB can be implemented in the end station (Talker/Listener) or within the nearest Bridge. The descriptions in this clause focus on Stream transformation in the end station and use features of the TSN user/network configuration information (46.2).

NOTE 1—In this clause, Stream transformation refers to changes to the fields of a frame that identify the Stream. IEEE Std 802.1CB specifies Stream transformation for identification as well as for frame replication and elimination (redundancy).

NOTE 2—Stream transformation is an optional capability of end stations and Bridges. If stream transformation is not supported, the user's identification of the Stream must be the same as the network's identification, and the user must use an identification that is consistent with bridging as specified for TSN features in this standard (e.g., VLAN tag and group destination MAC address).

Figure 46-4 provides an example of Stream transformation in the Talker end station. Stream transformation in the Listener end station is similar. The example of Figure 46-4 assumes use of the centralized network/ distributed user model (46.1.3.2). Use of the fully centralized model (46.1.3.3) is similar.

For Stream transformation in the end station, the end station’s interface to the network can provide the transformation capability, acting as a network entity within the user’s end station.

The identification of the Stream in the user originates from the User Application block, specified by the Talker as the DataFrameSpecification (46.2.3.4). The end station knows this user identification, but not the identification that the network requires. To negotiate the network identification, the Talker uses SRP to transmit InterfaceCapabilities (46.2.3.7) that describe its Stream transformation capabilities to the nearest Bridge. The Bridge uses MRP External Control (12.32.4) to send the InterfaceCapabilities to a CNC. The CNC consults its configuration of network identification and uses MRP External Control to send InterfaceConfiguration (46.2.5.3) along with successful Status (46.2.5) back to the Bridge. When the Bridge receives this information, it propagates back to the Talker using SRP. The InterfaceConfiguration provides the network identification, which the end station uses to perform Stream transformation for data frames.

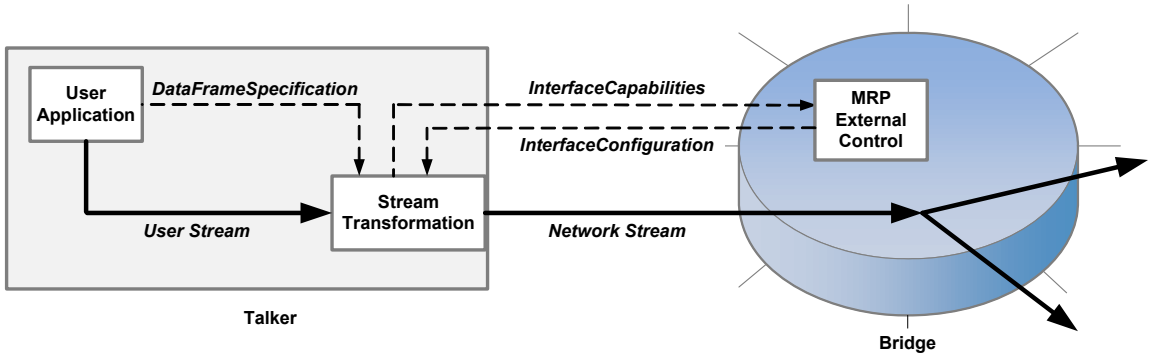

Figure 46-4—Example of Stream transformation in Talker end station

NOTE 3—The network identification typically entails allocation of a group MAC address for the Stream. If a CNC is used, the CNC can allocate a group MAC address from a pool that it maintains.

Figure 46-5 provides an example of IEEE 802.1CB functions within the Stream Transformation block in the Talker end station. The example assumes that the user identification uses an Internet Protocol (IP) packet for identification and that IP packet does not use the appropriate MAC address and VLAN tag for TSN features in Bridges (e.g., IP packet unicast destination MAC address, untagged). The IEEE 802.1CB function for IP Stream identification uses fields of the IP packet to identify the packet as a specific TSN stream. That stream identification is then applied to the IEEE 802.1CB function for Active Destination MAC and VLAN Stream identification to replace the destination MAC address and add a VLAN tag for TSN Bridge features. The IP fields of the packet are not changed. The IEEE 802.1CB functions can be implemented in software (e.g., operating system driver) or in hardware.

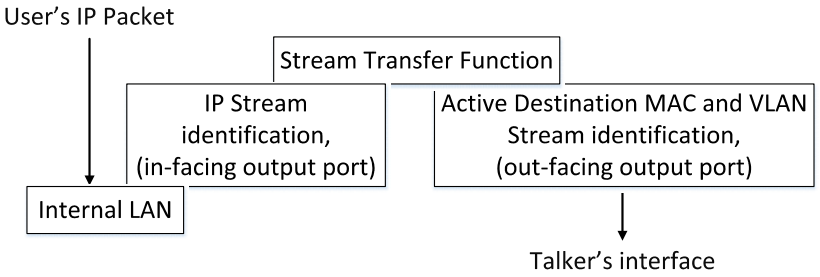

Figure 46-5—Example of IEEE 802.1CB functions in Talker end station

Figure 46-6 provides a corresponding example of IEEE 802.1CB functions within the Stream Transformation block in the Listener end station. In this direction, the IEEE 802.1CB function for Active Destination MAC and VLAN Stream identification provides both functions. The IEEE 802.1CB function uses the group destination MAC address and VLAN tag of the received frame to identify a specific Stream. The IEEE 802.1CB function then transforms the destination MAC address and VLAN tag to restore the Stream's frame to its original IP packet format (as transmitted by the Talker).

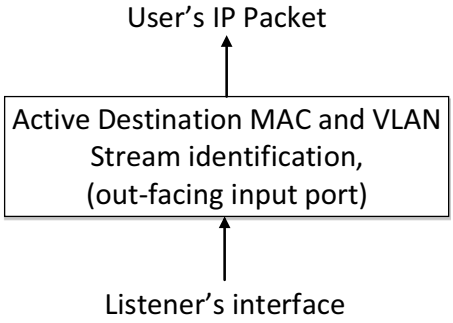

Figure 46-6—Example of IEEE 802.1CB functions in Listener end station

46.2 User/network configuration information

This subclause specifies the user/network configuration information that is used for the three TSN configuration models (46.1.3). The semantics for the TSN user/network configuration information is specified independent of schema, encoding, or protocol.

A schema or encoding of a protocol for the TSN user/network configuration information will reference 46.2 as part of its normative specifications. For the two distributed models, SRP specifies TLVs in 35.2.2.10 that reference information in 46.2. For the fully centralized model, the YANG module in 46.3 references information in 46.2.

Within this subclause, the word element refers to a single item of information used for TSN configuration. The word group refers to a collection of related elements. Groups are organized hierarchically, such that a group can be contained within another group. A single low-level group can be contained within multiple higher level groups. The dot-separated notation is used to refer to a specific element in text. For example, "Talker.StreamID.UniqueId" refers to the UniqueId element of the StreamID group that is contained within the Talker group.

This subclause specifies each group in a table. Each row of the table specifies an element of the group. Each element's row specifies its name, data type, and a reference to normative text that specifies its semantics (i.e., meaning). The data type of each element uses one of the values from 46.2.1. Other specifications such as conformance (i.e., required or optional), direction of transfer (i.e., user to network or network to user), default value, and range limitations are specified with normative text instead of with the table.

Each element name uses a camel-case naming convention (e.g., "MacAddress") to align with naming conventions used in other clauses of IEEE Std 802.1Q. A specific protocol can use a different naming convention (e.g., 46.3) as long as the protocol's name for the element can be associated with the element's specification in 46.2.

46.2.1 Data types

The data type of each element is limited to semantics, independent of a specific encoding or protocol. Data types in the tables include the following:

a) Boolean
b) int8, for a signed 8-bit integer
c) int16, for a signed 16-bit integer
d) int32, for a signed 32-bit integer

e) uint8, for an unsigned 8-bit integer
f) uint16, for an unsigned 16-bit integer
g) uint32, for an unsigned 32-bit integer
h) string
i) enumeration, for a collection of named values
j) rational, for a rational number consisting of a uint32 numerator and uint32 denominator
k) mac-address-type, for an IEEE 802 MAC address
l) ipv4-address-type, for an IPv4 address (IETF RFC 791 [B80])
m) ipv6-address-type, for an IPv6 address (IETF RFC 2460 [B81])
n) sequence of <X>, for a list of zero or more instances of data type <X> (e.g., sequence of uint32)

If the data type shown in the table is not from the preceding list, then the data type is specified in the normative text for the element.

46.2.2 Protocol integration

The TSN user/network configuration information consists of three high-level groups:

— **Talker:** Elements from user to network that specify the Talker for a single Stream.
— **Listener:** Elements from user to network that specify the Listener for a single Stream.
— **Status:** Elements from network to user that specify the status of the Stream's network configuration for a Talker or Listener. This group notifies the user when the Stream is ready for use (or a failure occurred).

A protocol that supports TSN user/network configuration information shall integrate the three groups using the specifications in this clause. The integration is provided in each protocol's specifications (not in this clause). For examples of such protocol integration, see Clause 35 and 46.3.

Each TSN configuration protocol shall use the StreamID of this clause (46.2.3.1) as the unique identifier of each Stream's configuration. The StreamID identifies configuration, not data, so it has no formal relation to the data frame encoding for the Stream.

TSN configuration can be viewed conceptually as a request/response exchange:

— Request: End station or CUC transmits a protocol message that contains a Talker or Listener group.
— Response: Bridge or CNC transmits a protocol message that contains a Status group.

The StreamID of each group correlates each request to its corresponding response, such that configuration of multiple Streams can overlap in time. For example, a possible sequence of messages is Talker request for StreamID 1, Talker request for StreamID 2, Listener request for StreamID 2, Status to Talker for StreamID 2, Status to Listener for StreamID 2, Listener request for StreamID 1, and so on.

For the fully distributed model (46.1.3.1), the Status response to a Talker is communicated in a message along with the Listener request(s). When there is more than one Listener per Stream, the Listener and Status groups must be merged as they propagate through Bridges to the Talker. The methodology for this merger is specified in the TSN configuration protocol (not in this clause).

For the fully centralized model (46.1.3.3), the CUC combines multiple Talker groups and Listener groups into a single transfer from CUC to CNC for computation together in the CNC. After the CNC computation is complete, a single transfer from CNC to CUC combines multiple Status groups. Each Status group can provide sub-groups for each Talker/Listener and merged groups for the entire Stream. The methodology for these combinations and mergers is specified in the TSN configuration protocol (not in this clause).

In addition to the request/response exchange, end stations can transmit Talker/Listener messages as advertisement (e.g., for a Talker to notify potential Listeners of its presence). The methodology for this advertisement is specified in the TSN configuration protocol (not in this clause).

For each Talker group and Listener group, there are two operations for each Stream:

— **Join:** Talker/Listener request to join the Stream and to configure network resources for this Stream's data transfer.
— **Leave:** Talker/Listener request to leave the Stream and to release the network resources for this Stream.

The groups in this clause specify requirements for the join and leave operation, but the encoding of the operation is not specified in this clause. Each TSN configuration protocol shall specify the encoding of the join/leave operation as part of its integration. For example, SRP specifications of this standard (Clause 35) encode the join/leave operation within the Multiple Registration Protocol (MRP) (Clause 10) that serves as its foundation.

The protocol message(s) that invoke the join or leave operation are not required to coincide with the protocol message(s) that contain the associated groups (Talker, Listener, or Status). Nevertheless, the groups specify elements that are required for a subsequent join or leave operation to be valid. For example, for the fully centralized model (46.1.3.3), the CUC can transfer a list of Talker/Listener groups to the CNC, followed by a separate protocol message with a join request that applies to the entire list. For the join request to succeed, each of the Talker/Listener groups must contain the required elements. At a later time, the CUC can read the resulting list of Status groups from the CNC, which provides the response to the join.

46.2.3 Talker

The Talker group specifies the following:

— Talker's behavior for Stream (how/when transmitted)
— Talker's requirements from the network
— TSN capabilities of the Talker's interface(s)

In the fully distributed model and the centralized network/distributed user model, this group originates from the Talker's end station. In the fully centralized model, this group originates from the CUC.

The Talker group contains the following groups:

— StreamID (46.2.3.1)
— StreamRank (46.2.3.2)
— EndStationInterfaces (46.2.3.3)
— DataFrameSpecification (46.2.3.4)
— TrafficSpecification (46.2.3.5)
— UserToNetworkRequirements (46.2.3.6)
— InterfaceCapabilities (46.2.3.7)

For the join operation, the StreamRank and TrafficSpecification groups shall be included within the Talker group.

For the join operation, the UserToNetworkRequirements and InterfaceCapabilities groups may be included within the Talker group. If a group is omitted, the network uses the default values specified for elements of that group.

For the join operation, the DataFrameSpecification shall be included within the Talker group unless all of the following conditions apply:

a) The InterfaceCapabilities group is included.
b) InterfaceCapabilities.CB-StreamIdenTypeList includes the type for Active Destination MAC and VLAN Stream identification.

The preceding conditions enable the Talker and Listener to perform Stream transformation (46.1.4), and therefore the network does not need the user's data frame identification in DataFrameSpecification. The absence of the DataFrameSpecification group notifies the network that Stream transformation is performed in the end stations for this Stream.

For the join and leave operation, the StreamID group shall be included in the TSN user/network protocol in a location associated with the Stream (e.g., Talker group).

For the join and leave operation, EndStationInterfaces shall be included within the Talker group for the fully centralized model (46.1.3.3). For the join and leave operation, EndStationInterfaces should be included within the Talker group for the fully distributed model (46.1.3.1) or centralized network/distributed user model (46.1.3.2).

46.2.3.1 StreamID

The StreamID group provides a unique identifier of the Stream's configuration and is used by protocols in the network to associate the user's Stream with TSN resources. The StreamID group is used by Talker, Listener, and Status groups.

The elements of the StreamID group are listed in Table 46-1.

Table 46-1—StreamID elements

Name	Data type	Reference
MacAddress	mac-address-type	46.2.3.1.1
UniqueID	uint16	46.2.3.1.2

46.2.3.1.1 MacAddress

MacAddress is a 48-bit IEEE 802 MAC address associated with the Talker sourcing the Stream to the bridged network. The entire range of MAC addresses are acceptable.

NOTE 1—The MAC address component of the StreamID can, but does not necessarily, have the same value as the source_address parameter of any frame in the actual data Stream. For example, the StreamID can be assigned by a TSN CUC (see 46.1.3.3), using a pool of MAC addresses that the TSN CUC maintains.

NOTE 2—If the MAC addresses used to construct StreamIDs are not unique within the network, duplicate StreamIDs can be generated, with unpredictable results.

46.2.3.1.2 UniqueID

UniqueID is used to distinguish between multiple Streams within the end station identified by MacAddress.

46.2.3.2 StreamRank

The StreamRank group provides the rank of this Stream relative to other Streams in the network. This rank is used to determine success/failure of Stream resource configuration and is unrelated to the Stream's data.

The elements of the StreamRank group are listed in Table 46-2.

Table 46-2—StreamRank elements

Name	Data type	Reference
Rank	uint8	46.2.3.2.1

46.2.3.2.1 Rank

The Rank is used by the network to decide which Streams can and cannot exist when TSN resources reach their limit. If a Bridge's Port becomes oversubscribed (e.g., network reconfiguration, IEEE 802.11 bandwidth reduction), the Rank is used to help determine which Streams can be dropped (i.e., removed from Bridge configuration).

The only valid values for Rank shall be zero and one. The configuration of a Stream with Rank zero is more important than the configuration of a Stream with Rank one. The Rank value of zero is intended for emergency traffic, and the Rank value of one is intended for non-emergency traffic.

NOTE—It is expected that higher layer applications and protocols can use the Rank to indicate the relative importance of Streams based on user preferences. Those user preferences are expressed by means beyond the scope of this standard. When multiple applications exist in a network (e.g., audio/video along with industrial control), it can be challenging for the varied applications and vendors to agree on multiple Rank values. To mitigate such challenges, this Rank uses a simple concept of emergency (zero) and non-emergency (one) that can be applied over all applications. For example, in a network that carries audio Streams for fire safety announcements, all applications are likely to agree that those Streams use Rank of zero.

46.2.3.3 EndStationInterfaces

The EndStationInterfaces group provides a list of one or more InterfaceID groups. Each InterfaceID group identifies a physical interface (distinct point of attachment) in the end station acting as a Talker/Listener.

Although many end stations contain a single interface, this list allows for multiple interfaces. Some TSN features allow a single Stream to span multiple interfaces (e.g., seamless redundancy).

In centralized network models, EndStationInterfaces is used by the CNC to locate the interfaces of the Talker/Listener in the topology.

The InterfaceID group is used to identify a distinct point of attachment in EndStationInterfaces, InterfaceConfiguration (46.2.5.3), and FailedInterfaces (46.2.5.4). In the context of EndStationInterfaces and InterfaceConfiguration, each interface is located in an end station only. In the context of FailedInterfaces, each interface can be located within an end station or a Bridge.

The elements of the InterfaceID group are listed in Table 46-3.

Table 46-3—InterfaceID elements

Name	Data type	Reference
MacAddress	mac-address-type	46.2.3.3.1
InterfaceName	string	46.2.3.3.2

46.2.3.3.1 MacAddress

MacAddress is the EUI-48 MAC address (IEEE Std 802) of the interface in the station (end station or Bridge). This MAC address uniquely identifies the station within the local network.

MacAddress shall be included in the InterfaceID group.

NOTE—This MAC address can be discovered in the physical topology using protocols such as IEEE Std 802.1AB (LLDP). LLDP supports MAC address as a subtype for the station's Chassis ID and Port ID. If the station does not use MAC address for its LLDP IDs, remote management can be used to associate this MacAddress to the values provided in the LLDP IDs.

46.2.3.3.2 InterfaceName

InterfaceName is the name of the interface that is assigned locally by the station (end station or Bridge).

InterfaceName may be included in the InterfaceID group.

IEEE Std 802 recommends that each distinct point of attachment to an IEEE 802 network have its own EUI MAC address. If the identified station follows this IEEE 802 recommendation, the MacAddress element uniquely identifies the interface as well as the station, and InterfaceName is not needed.

If the MacAddress applies to more than one interface (distinct point of attachment) within the station, InterfaceName provides a locally assigned name that can help to identify the interface.

When YANG is used for management of the station, InterfaceName is the interface name that serves as the key for the station's interface list (IETF RFC 7223 [B84]).

NOTE 1—The TSN CNC is typically located in a different physical product than the station identified by this InterfaceID. Since the InterfaceName is assigned locally by the identified station, it is possible that the station's product will change InterfaceName in a manner that the TSN CNC cannot detect. For example, IETF RFC 7223 [B84] mentions that the YANG interface name can change when a physical attachment point is inserted or removed.

NOTE 2—This interface name can be discovered in the physical topology using protocols such as IEEE Std 802.1AB (LLDP). LLDP supports interface name as a subtype for its Port ID. If the station does not use interface name for its LLDP Port ID, remote management can be used to associate this InterfaceName to the values provided in the LLDP Port ID.

46.2.3.4 DataFrameSpecification

The DataFrameSpecification group specifies the frame that carries the Talker's Stream data. The network uses the specification to identify this Stream's frames as TSN in order to apply the required TSN configuration.

The DataFrameSpecification is based on the user's knowledge of the frame, without any network specifics. In other words, this specifies the frame that the Talker would use in the absence of TSN.

The DataFrameSpecification is provided as a list of fields that the user knows. Each field is provided as one of the following groups:

— IEEE802-MacAddresses (46.2.3.4.1)
— IEEE802-VlanTag (46.2.3.4.2)
— IPv4-tuple (46.2.3.4.3)
— IPv6-tuple (46.2.3.4.4)

The list of fields is ordered from start of frame to end of header. For example, if the Talker uses a VLAN-tagged Ethernet frame (not IP), the list consists of IEEE802-MacAddresses followed by IEEE802-VlanTag. For example, if the Talker uses a UDP/IPv4 packet without knowledge of the Ethernet header, the list consists of IPv4-tuple only.

This group is optional, and its absence indicates that Stream transformation is performed in the Talker and Listeners of this Stream (46.2.3).

46.2.3.4.1 IEEE802-MacAddresses

The IEEE802-MacAddresses group specifies a pair of IEEE 802 MAC addresses for Stream identification.

The use of these fields for Stream identification corresponds to the managed objects for Stream identification in IEEE Std 802.1CB. If inconsistency arises between this specification and IEEE Std 802.1CB, IEEE Std 802.1CB takes precedence.

The elements of the IEEE802-MacAddresses group are listed in Table 46-4.

Table 46-4—IEEE802-MacAddresses elements

Name	Data type	Reference
DestinationMacAddress	mac-address-type	46.2.3.4.1 item a)
SourceMacAddress	mac-address-type	46.2.3.4.1 item b)

The elements of the IEEE802-MacAddresses group:

a) **DestinationMacAddress:** This field specifies a destination MAC address. An address of all 1's specifies that the destination MAC address is ignored for purposes of Stream identification.
b) **SourceMacAddress:** This field specifies a source MAC address. An address of all 1's specifies that the source MAC address is ignored for purposes of Stream identification.

46.2.3.4.2 IEEE802-VlanTag

The IEEE802-VlanTag group specifies a customer VLAN Tag (C-TAG of Clause 9) for Stream identification.

The Drop Eligible Indicator (DEI) field is not relevant from the perspective of a TSN Talker/Listener.

The use of these fields for Stream identification corresponds to the managed objects for Stream identification in IEEE Std 802.1CB. If inconsistency arises between this specification and IEEE Std 802.1CB, IEEE Std 802.1CB takes precedence.

The elements of the IEEE802-VlanTag group are listed in Table 46-5.

Table 46-5—IEEE802-VlanTag elements

Name	Data type	Reference
PriorityCodePoint	uint8	46.2.3.4.2 item a)
VlanId	uint16	46.2.3.4.2 item b)

The elements of the IEEE802-VlanTag group:

a) **PriorityCodePoint:** This field specifies the Priority Code Point (PCP) field of the VLAN tag. The value range is 0 to 7 inclusive. The PriorityCodePoint is not used to identify the Stream, but it does identify a traffic class (queue) in Bridges.
b) **VlanId:** This field specifies the VLAN ID (VID) field of the VLAN tag. The value range is 0 to 4095 inclusive. If only the PriorityCodePoint is known, the VlanId is specified as 0.

46.2.3.4.3 IPv4-tuple

The IPv4-tuple group specifies fields to identify an IPv4 (RFC791) Stream.

The use of these fields for Stream identification corresponds to the managed objects for Stream identification in IEEE Std 802.1CB. If inconsistency arises between this specification and IEEE Std 802.1CB, IEEE Std 802.1CB takes precedence.

The elements of the IPv4-tuple group are listed in Table 46-6.

Table 46-6—IPv4-tuple elements

Name	Data type	Reference
SourceIpAddress	ipv4-address-type	46.2.3.4.3 item a)
DestinationIpAddress	ipv4-address-type	46.2.3.4.3 item b)
Dscp	uint8	46.2.3.4.3 item c)
Protocol	uint16	46.2.3.4.3 item d)
SourcePort	uint16	46.2.3.4.3 item e)
DestinationPort	uint16	46.2.3.4.3 item f)

The elements of the IPv4-tuple group:

a) **SourceIpAddress:** This field specifies the source IPv4 address. An address of all 0's specifies that the IP source address is ignored for purposes of Stream identification.
b) **DestinationIpAddress:** This field specifies the destination IPv4 address.

c) **Dscp:** This field specifies the Differentiated Services Code Point (DSCP) (IETF RFC 2474). A value of 64 decimal specifies that the DSCP is ignored for purposes of Stream identification.
d) **Protocol:** This field specifies the IPv4 Protocol (e.g., UDP). The special value of all 1's (FFFF hex) represents 'None', i.e., Protocol, SourcePort, and DestinationPort are ignored for purposes of Stream identification. For any value other than all 1's, the lower octet is used to match IPv4 Protocol.
e) **SourcePort:** This field specifies the source port of the Protocol.
f) **DestinationPort:** This field specifies the destination port of the Protocol.

46.2.3.4.4 IPv6-tuple

The IPv6-tuple group specifies fields to identify an IPv6 (RFC2460) Stream.

The use of these fields for Stream identification corresponds to the managed objects for Stream identification in IEEE Std 802.1CB. If inconsistency arises between this specification and IEEE Std 802.1CB, IEEE Std 802.1CB takes precedence.

The elements of the IPv6-tuple group are listed in Table 46-7.

Table 46-7—IPv6-tuple elements

Name	Data type	Reference
SourceIpAddress	ipv6-address-type	46.2.3.4.3 item a)
DestinationIpAddress	ipv6-address-type	46.2.3.4.3 item b)
Dscp	uint8	46.2.3.4.3 item c)
Protocol	uint16	46.2.3.4.3 item d)
SourcePort	uint16	46.2.3.4.3 item e)
DestinationPort	uint16	46.2.3.4.3 item f)

The elements of the IPv6-tuple group:

a) **SourceIpAddress:** This field specifies the source IPv6 address. An address of all 0's specifies that the IP source address is ignored for purposes of Stream identification.
b) **DestinationIpAddress:** This field specifies the destination IPv6 address.
c) **Dscp:** This field specifies the DSCP (IETF RFC 2474). A value of 64 decimal specifies that the DSCP is ignored for purposes of Stream identification.
d) **Protocol:** This field specifies the IPv6 Next Header (e.g., UDP). The special value of all 1's (FFFF hex) represents 'None', i.e., Protocol, SourcePort, and DestinationPort are ignored for purposes of Stream identification. For any value other than all 1's, the lower octet is used to match IPv6 Next Header.
e) **SourcePort:** This field specifies the source port of the Protocol.
f) **DestinationPort:** This field specifies the destination port of the Protocol.

46.2.3.5 TrafficSpecification

The TrafficSpecification group specifies how the Talker transmits frames for the Stream. This is effectively the Talker's promise to the network. The network uses this traffic specification to allocate resources and adjust queue parameters in Bridges.

The TrafficSpecification group consists of the following:

a) Its required elements, listed in Table 46-8.

Table 46-8—TrafficSpecification elements

Name	Data type	Reference
Interval	rational	46.2.3.5.1
MaxFramesPerInterval	uint16	46.2.3.5.2
MaxFrameSize	uint16	46.2.3.5.3
TransmissionSelection	uint8	46.2.3.5.4

b) An optional TSpecTimeAware group, whose elements are listed in Table 46-9.

Table 46-9—TSpecTimeAware elements

Name	Data type	Reference
EarliestTransmitOffset	uint32	46.2.3.5.5
LatestTransmitOffset	uint32	46.2.3.5.6
Jitter	uint32	46.2.3.5.7

The presence of TSpecTimeAware specifies that the Talker's traffic is synchronized to a known time on the network (e.g., using IEEE Std 802.1AS). The TSpecTimeAware group provides elements to specify the Talker's time-aware transmit to the network. The Talker and Listeners of a Stream are assumed to coordinate using user (application) mechanisms, such that each Listener is aware that its Talker transmits in a time-aware manner. If MaxFramesPerInterval is greater than one, the time-aware Talker shall transmit multiple frames as a burst within the Interval, with the minimum inter-frame gap allowed by the media.

NOTE—Although scheduled traffic (8.6.8.4) specifies a valid implementation of a time-aware Talker, the TSpecTimeAware group is intended to support alternate implementations of scheduling.

For informative examples using the TSpecTimeAware group of TrafficSpecification, refer to U.1.

46.2.3.5.1 Interval

Interval specifies the period of time in which the traffic specification cannot be exceeded. The traffic specification is specified by MaxFramesPerInterval and MaxFrameSize.

The Interval is a rational number of seconds, defined by an unsigned 32-bit integer numerator and an unsigned 32-bit integer denominator.

If the TSpecTimeAware group is not present, the Interval specifies a sliding window of time. The Talker's transmission is not synchronized to a time on the network, and therefore the traffic specification cannot be exceeded during any Interval in time.

If the TSpecTimeAware group is present, the Interval specifies a window of time that is aligned with the time epoch that is synchronized on the network. For example, if IEEE Std 802.1AS-2011 [B11] is used with the PTP timescale, the first Interval begins at 1 January 00:00:00 TAI. If CurrentTime represents the current time, then the start of the next Interval (StartOfNextInterval) is

StartOfNextInterval = N × Interval

where N is the smallest integer for which the relation

StartOfNextInterval ≥ CurrentTime

would be TRUE.

46.2.3.5.2 MaxFramesPerInterval

MaxFramesPerInterval specifies the maximum number of frames that the Talker can transmit in one Interval.

46.2.3.5.3 MaxFrameSize

MaxFrameSize specifies maximum frame size that the Talker will transmit, excluding any overhead for media-specific framing (e.g., preamble, IEEE 802.3 header, Priority/VID tag, CRC, interframe gap). As the Talker or Bridge determines the amount of bandwidth to reserve on the egress Port (interface), it will calculate the media-specific framing overhead on that Port and add it to the number specified in the MaxFrameSize element.

46.2.3.5.4 TransmissionSelection

TransmissionSelection specifies the algorithm that the Talker uses to transmit this Stream's traffic class. This algorithm is often referred to as the shaper for the traffic class.

The value for this element uses Table 8-6 (in 8.6.8). If no algorithm is known, the value zero (strict priority) can be used.

The Talker's shaping and scheduling of the Stream is considered to be on the user side of the user/network boundary, and this specifies the Talker's behavior to the network. In the fully distributed model, this value can be ignored by a Bridge, and it is not changed during propagation through the Bridge.

46.2.3.5.5 EarliestTransmitOffset

EarliestTransmitOffset specifies the earliest offset within the Interval at which the Talker is capable of starting transmit of its frames. As part of the Status group, the network will return a specific TimeAwareOffset to the Talker (within the earliest/latest range), which the Talker uses to schedule its transmit.

EarliestTransmitOffset is specified as an integer number of nanoseconds.

The Talker's transmit offsets include EarliestTransmitOffset, LatestTransmitOffset, and the TimeAwareOffset returned to the Talker in the Status group. Each of the Talker's offsets is specified at the point when the message timestamp point of the first frame of the Stream passes the reference plane marking the boundary between the network media and PHY. The message timestamp point is specified by IEEE Std 802.1AS for various media.

46.2.3.5.6 LatestTransmitOffset

LatestTransmitOffset specifies the latest offset within the Interval at which the Talker is capable of starting transmit of its frames. As part of the Status group, the network will return a specific TimeAwareOffset to the Talker (within the earliest/latest range), which the Talker uses to schedule its transmit.

LatestTransmitOffset is specified as an integer number of nanoseconds.

46.2.3.5.7 Jitter

Jitter specifies the maximum difference in time between the Talker's transmit offsets and the ideal synchronized network time (e.g., IEEE 802.1AS time). Jitter is specified as an unsigned integer number of nanoseconds.

The maximum difference means sooner or later than the ideal (e.g., transmit ± 500 ns relative to IEEE 802.1AS time results in Jitter of 500).

The ideal synchronized network time refers to time at the source (e.g., IEEE 802.1AS grandmaster). The Jitter does not include inaccuracies as time is propagated from the time source to the Talker because those inaccuracies are assumed to be known by the network and time synchronization is a network technology. The Jitter element is intended to specify inaccuracies in the Talker's implementation. For example, if the Talker's IEEE 802.1AS time is ± 812 ns relative to the grandmaster and the Talker schedules using a 100 µs timer tick driven by IEEE 802.1AS time, Jitter is 50 000 (not 50 812).

The Talker's transmit offsets include EarliestTransmitOffset, LatestTransmitOffset, and the TimeAwareOffset returned to the Talker in the Status group.

46.2.3.6 UserToNetworkRequirements

The UserToNetworkRequirements group specifies user requirements for the Stream, such as latency and redundancy.

The network (e.g., CNC) will merge all UserToNetworkRequirements for a Stream to ensure that all requirements are met.

The elements of the UserToNetworkRequirements group are listed in Table 46-10.

Table 46-10—UserToNetworkRequirements elements

Name	Data type	Reference
NumSeamlessTrees	uint8	46.2.3.6.1
MaxLatency	uint32	46.2.3.6.2

46.2.3.6.1 NumSeamlessTrees

NumSeamlessTrees specifies the number of trees that the network will configure to deliver seamless redundancy for the Stream.

The value zero is interpreted as one (i.e., no seamless redundancy).

This requirement is provided from the Talker only. Listeners shall set this element to one.

From each Talker to a single Listener, the network configures a path that relays Stream data through Bridges. If the Talker has more than one Listener, the network configures a tree of multiple paths.

NumSeamlessTrees specifies the number of maximally disjoint trees that the network shall configure from the Talker to all Listeners. Each tree is disjoint from other trees, in that the network evaluates the physical topology to avoid sharing the same Bridge and links in each tree's paths. This computation of disjoint trees is maximal, in that shared Bridges and links are avoided to the maximum extent allowed by the physical topology. For example, if a single link exists from a Bridge to a Listener and NumSeamlessTrees is three, then all three trees will share that link to the Listener.

When NumSeamlessTrees is greater than one, the transfer of the Stream's data frames shall use a seamless redundancy standard, such as IEEE Std 802.1CB. When a link shared by multiple trees diverges to multiple disjoint links, the seamless redundancy standard replicates (i.e., forwards a distinct copy of each data frame to the disjoint trees). When disjoint trees converge to a single link, the seamless redundancy standard eliminates the duplicate copies of each data frame. Assuming that other sources of frame loss are mitigated (e.g., congestion), failure of a link or Bridge in one disjoint tree does not result in frame loss as long as at least one remaining disjoint tree is operational.

If the Talker sets this element to one, the network may make use of redundancy standards that are not seamless (i.e., failure of link results in lost frames), such as MSTP and IS-IS.

If the Talker sets this element to greater than one and seamless redundancy is not possible in the current network (no disjoint paths or no seamless redundancy standard in Bridges), the Status group returns a nonzero FailureCode (46.2.5.1).

If the UserToNetworkRequirements group is not provided within the Talker or Listener group, the network shall use the default value of one for this element.

46.2.3.6.2 MaxLatency

MaxLatency specifies the maximum latency from Talker to Listener(s) for a single frame of the Stream.

MaxLatency is specified as an integer number of nanoseconds.

Latency shall use the definition of 3.118, with additional context as follows: The "known reference point in the frame" is the message timestamp point specified in IEEE Std 802.1AS for various media (i.e., start of the frame). The "first point" is in the Talker at the reference plane marking the boundary between the network media and PHY (see IEEE Std 802.1AS). The "second point" is in the Listener at the reference plane marking the boundary between the network media and PHY.

When this requirement is specified by the Talker, it must be satisfied for all Listeners.

When this requirement is specified by the Listener, it must be satisfied for this Listener only.

If the UserToNetworkRequirements group is not provided within the Talker or Listener group, the network shall use the default value zero for this element.

The special value of zero represents usage of the initial value of Status.AccumulatedLatency as the maximum latency requirement. This effectively locks down the initial latency that the network calculates after successful configuration of the Stream, such that any subsequent increase in latency beyond that value causes the Stream to fail.

The assumption for when the "first point" occurs in the Talker depends on the presence of the TSpecTimeAware group in the Talker's TrafficSpecification.

a) When TSpecTimeAware is not present
   1) The Talker is assumed to transmit at an arbitrary time (not scheduled).
b) When TSpecTimeAware is present
   1) The 'first point' is assumed to occur at the start of the Interval as if the Talker's offsets (EarliestTransmitOffset and LatestTransmitOffset of 46.2.3.5) are both zero. The Talker's offsets are not typically zero, but use of the start of Interval for purposes of MaxLatency allows the Listener(s) to schedule their application independently from the Talker's offset configuration.
   2) The Listener determines MaxLatency based on its scheduling of a read function in the application. Nevertheless, the time from frame reception (i.e., "second point") to execution of the read function is in the user's scope and therefore not included in MaxLatency.
   3) MaxLatency can be set to a value greater than the Talker's Interval in order to specify a longer latency requirement. For example, if the Talker's Interval is 500 μs and MaxLatency is 700 μs, the Listener receives the frame no later than 200 μs into the interval that follows the Talker's Interval.

46.2.3.7 InterfaceCapabilities

The InterfaceCapabilities group specifies the network capabilities of all interfaces (Ports) contained in the EndStationInterfaces group.

The network may provide configuration of these capabilities in the InterfaceConfiguration group of the Status group.

NOTE—If an end station contains multiple interfaces with different network capabilities, each interface should be specified in a distinct Talker or Listener group (i.e., one interface in EndStationInterfaces). Use of multiple interface in EndStationInterfaces is intended for network capabilities that span multiple interfaces (e.g., seamless redundancy).

The elements of the InterfaceCapabilities group are listed in Table 46-11.

Table 46-11—InterfaceCapabilities elements

Name	Data type	Reference
VlanTagCapable	Boolean	46.2.3.7.1
CB-StreamIdenTypeList	sequence of uint32	46.2.3.7.2
CB-SequenceTypeList	sequence of uint32	46.2.3.7.3

46.2.3.7.1 VlanTagCapable

When VlanTagCapable is true, the interface supports the ability to tag/untag frames using a Customer VLAN Tag (C-TAG of Clause 9) provided by the network.

For a Talker, the network's tag replaces the tag specified by the DataFrameSpecification. If the DataFrameSpecification is untagged (no IEEE802-VlanTag group), the network's tag is inserted in the frame as it passes through the interface.

For a Listener, the user's tag from the DataFrameSpecification replaces the network's tag as the frame passes through the interface. If the DataFrameSpecification is untagged (no IEEE802-VlanTag group), the network's tag is removed from the frame as it passes through the interface.

If the end station supports more than one interface (i.e., more than one entry in EndStationInterfaces), VlanTagCapable of true means that a distinct VLAN tag can be applied to each interface. The list of VLAN tags (one for each interface) can be provided by the network in InterfaceConfiguration.InterfaceList (IEEE802-VlanTag choice).

When VlanTagCapable is false, the interface does not support the capability to tag/untag frames using a Customer VLAN Tag (C-TAG of Clause 9) provided by the network.

If the InterfaceCapabilities group is not provided within the Talker or Listener group, the network shall use the default value of false for this element.

46.2.3.7.2 CB-StreamIdenTypeList

CB-StreamIdenTypeList provides a list of the supported Stream Identification types as specified in IEEE Std 802.1CB.

Each Stream Identification type is provided as a 32-bit unsigned integer. The upper 3 octets contain the OUI/CID, and the lowest octet contains the type number.

NOTE—If the Talker/Listener end system supports IEEE Std 802.1CB, Null Stream identification is required, and that Stream Identification type is included in this list. If the Talker/Listener end system does not support IEEE Std 802.1CB, this list is empty.

If the end station supports more than one interface (i.e., more than one InterfaceID in EndStationInterfaces), an empty CB-StreamIdenTypeList means that the end station is capable of transferring the Stream on any one of its interfaces (not all). When this is specified, the network shall decide which interface is best used for TSN purposes and communicate that decision by returning a single interface in InterfaceConfiguration.InterfaceList. The Talker/Listener uses this interface alone for the Stream.

If the InterfaceCapabilities group is not provided within the Talker or Listener group, the network shall use an empty list as the default value for this element.

46.2.3.7.3 CB-SequenceTypeList

CB-SequenceTypeList provides a list of the supported Sequence Encode/Decode types as specified in IEEE Std 802.1CB.

Each sequence type is provided as a 32-bit unsigned integer. The upper 3 octets contain the OUI/CID, and the lowest octet contains the type number.

If the InterfaceCapabilities group is not provided within the Talker or Listener group, the network shall use an empty list as the default value for this element.

46.2.4 Listener

The Listener group specifies the following:

— Listener’s requirements from the network
— TSN capabilities of the Listener’s interface(s)

In the fully distributed model and the centralized network/distributed user model, this group originates from the Listener’s end station. In the fully centralized model, this group originates from the CUC.

The Listener group contains the following groups:

— StreamID (46.2.3.1)
— EndStationInterfaces (46.2.3.3)
— UserToNetworkRequirements (46.2.3.6)
— InterfaceCapabilities (46.2.3.7)

For the join operation, the UserToNetworkRequirements and InterfaceCapabilities groups may be included within the Listener group. If a group is omitted, the network uses the default values specified for elements of that group.

For the join and leave operation, the StreamID group shall be included in the TSN user/network protocol in a location associated with the Stream (e.g., Listener group).

For the join and leave operation, EndStationInterfaces shall be included within the Listener group for the fully centralized model (46.1.3.3). For the join and leave operation, EndStationInterfaces should be included within the Listener group for the fully distributed model (46.1.3.1) or centralized network/distributed user model (46.1.3.2).

46.2.5 Status

The Status group provides the status of a Stream's configuration from the network to each user (Talker or Listener).

In the fully distributed model and the centralized network/distributed user model, this group is delivered to each Talker end station and Listener end station of the Stream. In the fully centralized mode, this group is delivered to the CUC.

The Status group contains the following groups:

— StreamID (46.2.3.1)
— StatusInfo (46.2.5.1)
— AccumulatedLatency (46.2.5.2)
— InterfaceConfiguration (46.2.5.3)
— FailedInterfaces (46.2.5.4)

The AccumulatedLatency and InterfaceConfiguration groups are distinct for each Talker or Listener, and therefore Status is distinct.

The protocol for TSN configuration shall specify that one Status group is received in response to the transmit of a Talker group or Listener group. This provides at least one response to each join or leave request.

StreamID (46.2.3.1) identifies the Stream for which status is provided.

StatusInfo (46.2.5.1) provides the status of the Stream's configuration in the network.

For the join and leave operation, the StreamID and StatusInfo groups shall be included in the TSN user/network protocol in a location associated with the Stream (e.g., Status group).

For the join operation, the AccumulatedLatency group shall be included within the Status group that is sent to a Listener. For the join operation, the AccumulatedLatency group may be included within the Status group that is sent to a Talker.

For the join operation, the InterfaceConfiguration group may be included within the Status group. InterfaceConfiguration provides configuration of the interfaces in a specific end station.

If a group is omitted, the network uses the default values specified for elements of that group.

When a failure occurs in network configuration, FailedInterfaces identifies one or more interfaces (Ports) on which the failure occurred. If the FailureCode of the StatusInfo group (46.2.5.1) is zero, the FailedInterfaces group shall not be included within the Status group. If the FailureCode is nonzero, the FailedInterfaces group may be included within the Status group

When a Talker and at least one Listener have joined the stream and network resources are configured for the Stream (success or failure), the network shall transmit Status to the Talker and Listener(s). After the initial Status, upon any change in the status of the network configuration (e.g., new Listeners, change in FailureCode), the network shall transmit Status to the Talker and Listener(s). When the Stream does not have a Talker or a Listener, the network configuration is removed, and the network shall transmit a final Status to the remaining Talker and/or Listener(s). Beyond these requirements, additional transmit of Status is specified by the protocol that carries the Status group.

46.2.5.1 StatusInfo

The StatusInfo group provides information regarding the status of a Stream's configuration in the network.

The elements of the StatusInfo group are listed in Table 46-12.

Table 46-12—StatusInfo elements

Name	Data type	Reference
TalkerStatus	enumeration	46.2.5.1.1
ListenerStatus	enumeration	46.2.5.1.2
FailureCode	uint8	46.2.5.1.3

46.2.5.1.1 TalkerStatus

TalkerStatus provides the status of the network configuration for the Stream's Talker.

TalkerStatus uses the enumeration specified in Table 46-13.

Table 46-13—TalkerStatus enumeration

Name	Value	Description
None	0	No Talker detected.
Ready	1	Talker ready (configured).
Failed	2	Talker failed.

46.2.5.1.2 ListenerStatus

ListenerStatus provides the status of the network configuration for the Stream's Listener(s). If there is more than one Listener for the Stream, ListenerStatus provides the status of all Listeners.

ListenerStatus uses the enumeration specified in Table 46-14.

Table 46-14—ListenerStatus enumeration

Name	Value	Description
None	0	No Listener detected.
Ready	1	All Listeners ready (configured).
PartialFailed	2	One or more Listeners ready, and one or more Listeners failed. If Talker is ready, Stream can be used.
Failed	3	All Listeners failed.

46.2.5.1.3 FailureCode

If the Stream encounters a failure (TalkerStatus is Failed, or ListenerStatus is Failed, or ListenerStatus is PartialFailed), FailureCode provides a nonzero code that specifies the problem. If the Stream is configured without a failure, FailureCode is zero.

Table 46-15 specifies the nonzero values for FailureCode.

Table 46-15—TSN Failure Codes

Failure Code	Description of cause
1	Insufficient bandwidth
2	Insufficient Bridge resources
3	Insufficient bandwidth for traffic class
4	StreamID in use by another Talker
5	Stream destination_address already in use
6	Stream preempted by higher rank
7	Reported latency has changed
8	Egress Port is not AVB capable [a]
9	Use a different destination_address (i.e., MAC DA hash table full)
10	Out of MSRP resources
11	Out of MMRP resources
12	Cannot store destination_address (i.e., Bridge is out of MAC DA resources)
13	Requested priority is not an SR Class (3.259) priority

Table 46-15—TSN Failure Codes *(continued)*

Failure Code	Description of cause
14	MaxFrameSize [item a) in 35.2.2.8.4] is too large for media
15	msrpMaxFanInPorts [item f) in 35.2.1.4] limit has been reached
16	Changes in FirstValue, other than AccumulatedLatency, for a registered StreamID
17	VLAN is blocked or filtered on this egress Port [b]
18	VLAN tagging is disabled on this egress Port (untagged set)
19	SR class priority mismatch
20	Enhanced feature cannot be propagated to original Port
21	MaxLatency exceeded
22	Nearest Bridge cannot provide network identification for stream transformation
23	Stream transformation not supported
24	Stream identification type not supported for stream transformation
25	Enhanced feature cannot be supported without a CNC

[a] A device could choose to use the asCapable variable from 10.2.4.1 of IEEE Std 802.1AS-2011 [B11] to help determine if its neighboring device is AVB capable. If the asCapable variable is FALSE for a particular Port, then the neighboring device is not a time-aware system and therefore not AVB capable.
[b] This Failure Code is never declared in a Talker Failed message since Talker attributes are not propagated on egress Ports that have the associated VLAN blocked in the VLAN spanning tree (7.3) or filtered in VLAN Registration Entries (8.8). The Bridge can still be queried by other means to learn why the Talker attribute was not declared.

46.2.5.2 AccumulatedLatency

The AccumulatedLatency group provides the worst-case latency that a single frame of the Stream can encounter along its current path(s).

The elements of the AccumulatedLatency group are listed in Table 46-16.

Table 46-16—AccumulatedLatency elements

Name	Data type	Reference
AccumulatedLatency	uint32	46.2.5.2.1

46.2.5.2.1 AccumulatedLatency

The AccumulatedLatency element provides the worst-case maximum latency that a single frame of the Stream can encounter along its current path(s).

AccumulatedLatency is distinct for each Talker or Listener of the Stream.

When provided to a Listener, AccumulatedLatency is the worst-case maximum latency for that Listener only.

When provided to a Talker, AccumulatedLatency is the worst-case maximum latency for all Listeners (worst path).

AccumulatedLatency is specified as an integer number of nanoseconds.

AccumulatedLatency uses the same definition for latency as UserToNetworkRequirements.MaxLatency (46.2.3.6).

For a successful StatusInfo (46.2.5.1), the network returns a value less than or equal to UserToNetworkRequirements.MaxLatency.

If the TSpecTimeAware group is present in the TrafficSpecification group (46.2.3.5) of the Talker, the value is expressed as nanoseconds after the start of the Talker's Interval.

If the TSpecTimeAware group is not present in the TrafficSpecification group of the Talker, the value is expressed as nanoseconds after the Talker's transmit of any frame in the Stream, at any arbitrary time.

If UserToNetworkRequirements.NumSeamlessTrees is one, AccumulatedLatency shall provide the worst-case maximum latency for the current path from Talker to each Listener. If the path is changed (e.g., by a spanning tree protocol), AccumulatedLatency changes accordingly.

If UserToNetworkRequirements.NumSeamlessTrees is greater than one, AccumulatedLatency shall provide the worst-case maximum latency for all paths in use from the Talker to each Listener.

46.2.5.3 InterfaceConfiguration

The InterfaceConfiguration group provides configuration of interfaces in the Talker/Listener. This configuration assists the network in meeting the Stream's requirements. The InterfaceConfiguration meets the capabilities of the interface as provided in the InterfaceCapabilities group.

The InterfaceConfiguration group is distinct for each Talker or Listener of the Stream.

A distinct configuration is provided for each interface in the Talker/Listener (even if multiple interfaces use the same configuration). Each interface configuration consists of a single InterfaceID (46.2.3.3) group, followed by a list of configuration values for that interface.

The values in InterfaceID shall match one of the InterfaceID entries in the Talker/Listener EndStationInterfaces group.

The list of configuration values uses zero or more of the following groups:

— IEEE802-MacAddresses (46.2.5.3.1)
— IEEE802-VlanTag (46.2.5.3.2)
— IPv4-tuple (46.2.5.3.3)
— IPv6-tuple (46.2.5.3.4)
— TimeAwareOffset (46.2.5.3.5)

If the InterfaceConfiguration group is not provided within the Status group, the network shall assume zero configuration values as the default (no interface configuration).

46.2.5.3.1 IEEE802-MacAddresses

The IEEE802-MacAddresses group provides the source and destination MAC addresses that apply to the network side of the user/network boundary.

NOTE 1—On the user side, the MAC addresses are in DataFrameSpecification.IEEE802-MacAddresses.

NOTE 2—The source MAC address of the network is typically the same as the user. The destination MAC address can be different. For example, the user can use an individual address, but the network can use a group (multicast) address.

This group uses the specifications from DataFrameSpecification.IEEE802-MacAddresses (46.2.3.4.1).

This configuration value is not provided unless IEEE Std 802.1CB is supported and a value for Active Destination MAC and VLAN Stream identification is provided in CB-StreamIdenTypeList of InterfaceCapabilities.

46.2.5.3.2 IEEE802-VlanTag

The IEEE802-VlanTag group provides the Customer VLAN Tag (C-TAG of Clause 9) that applies to the network side of the user/network boundary.

NOTE—On the user side, the VLAN tag is in DataFrameSpecification.IEEE802-VlanTag (including untagged if that field is not provided).

This group uses the specifications from DataFrameSpecification.IEEE802-VlanTag (46.2.3.4.2).

If the user provides a VLAN ID in the IEEE802-VlanTag of DataFrameSpecification, the Stream's data frames are assumed to be limited to the active topology for that VLAN ID. Therefore, if the network uses a different VLAN ID in this configuration value, the network shall ensure that the replacement VLAN ID is limited to the equivalent active topology.

This configuration value is not provided unless VlanTagCapable of InterfaceCapabilities is true.

46.2.5.3.3 IPv4-tuple

The IPv4-tuple group provides the IPv4 identification that applies to the network side of the user/network boundary.

This group uses the specifications from DataFrameSpecification.IPv4-tuple (46.2.3.4.3).

This configuration value is not provided unless IEEE Std 802.1CB is supported and a value for IP Stream identification is provided in CB-StreamIdenTypeList of InterfaceCapabilities.

46.2.5.3.4 IPv6-tuple

The IPv6-tuple group provides the IPv6 identification that applies to the network side of the user/network boundary.

This group uses the specifications from DataFrameSpecification.IPv6-tuple (46.2.3.4.4).

This configuration value is not provided unless IEEE Std 802.1CB is supported and a value for IP Stream identification is provided in CB-StreamIdenTypeList of InterfaceCapabilities.

46.2.5.3.5 TimeAwareOffset

If the TSpecTimeAware group is present in the TrafficSpecification group (46.2.3.5) of the Talker, this configuration value shall be provided by the network to the Talker.

If the TSpecTimeAware group is not present in the TrafficSpecification group (46.2.3.5) of the Talker, this configuration value shall not be provided by the network.

This configuration value shall not be provided to Listeners as it is not applicable.

TimeAwareOffset specifies the offset that the Talker shall use for transmit. The network returns a value between EarliestTransmitOffset and LatestTransmitOffset of the Talker's TrafficSpecification. The value is expressed as nanoseconds after the start of the Talker's Interval. The data type is uint32.

46.2.5.4 FailedInterfaces

When a failure occurs in network configuration (i.e., nonzero FailureCode in StatusInfo group), FailedInterfaces provides a list of one or more physical interfaces (distinct points of attachment) in the failed end station or Bridge. Each identifier is sufficient to locate the interfaces in the physical topology.

The FailedInterfaces group is optional.

FailedInterfaces consists of a sequence of zero or more entries, each entry using the InterfaceID group specified in 46.2.3.3.

46.3 YANG data module definitions for TSN user/network configuration

In order to support the use of YANG-based protocols for the fully centralized model (46.1.3.3), 46.3.1 specifies a YANG module.

If a YANG-based protocol is specified by another standard for the TSN user/network configuration information (46.2), that specification shall use the YANG module specified in 46.3.1 [see item d) in 5.29].

The YANG module of 46.3.1 provides YANG text for each group of elements in 46.2. Each element is specified using a YANG `leaf`. Each group is specified as a YANG `typedef` or `grouping`. The YANG module for user/network configuration imports the YANG module of 46.3.1 and uses the `typedef` and `grouping` nodes in order to specify the schema tree used for communication between CUC and CNC.

YANG identifiers use a naming convention of hyphens between lowercase names (e.g., "mac-address"). Identifiers for elements and groups in 46.2 use a naming convention of camel case (e.g., "MacAddress"). The specifications for an identifier in 46.3.1 shall be interpreted as applying to the corresponding identifier in 46.2 regardless of differences in naming convention (e.g., requirements for "MacAddress" in 46.2 apply to "mac-address" in 46.3.1).

In the YANG module definitions of 46.3.1, if any discrepancy between the "description" text and the corresponding specifications in 46.2 occurs, the specifications in 46.2 take precedence.

46.3.1 Definition for the ieee802-dot1q-tsn-types YANG module

```
module ieee802-dot1q-tsn-types {

  namespace "urn:ieee:std:802.1Q:yang:ieee802-dot1q-tsn-types";
  prefix "dot1q-tsn-types";
```

```
  import ietf-inet-types { prefix "inet"; }

  organization
    "Institute of Electrical and Electronics Engineers";

  contact
    "WG-URL: http://ieee802.org/1/
    WG-EMail: stds-802-1@ieee.org

    Contact: IEEE 802.1 Working Group Chair
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            Piscataway
            NJ 08854
            USA

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

  description
    "Common typedefs and groupings for TSN user/network configuration
    in IEEE Std 802.1Q.";

  revision 2018-02-15 {
    description
      "Initial revision specified in 46.3 of IEEE Std 802.1Qcc-2018,
      Amendment: Stream Reservation Protocol (SRP) Enhancements
      and Performance Improvements.";
    reference
      "46.3 of IEEE Std 802.1Qcc-2018";
  }

  typedef stream-id-type {
    type string {
      pattern '[0-9a-fA-F]{2}(-[0-9a-fA-F]{2}){5}:[0-9a-fA-F]{2}-[0-9a-
fA-F]{2}';
    }
    description
      "This typedef specifies a Stream ID, a unique identifier
      of the Stream's configuration, used by protocols in the
      network to associate the user's Stream with TSN resources.

      The Stream ID is a string that represents two fields:

      MAC Address:

        A 48-bit IEEE 802 MAC address associated with
        the Talker sourcing the Stream to the bridged network.
        The entire range of MAC addresses are acceptable.

        NOTE 1—The MAC address component of the StreamID can,
        but does not necessarily, have the same value as the
        source_address parameter of any frame in the actual
```

```
        data Stream. For example, the Stream ID can be assigned
        by a TSN CUC (see 46.1.3.3 of IEEE Std 802.1Qcc-2018),
        using a pool of MAC addresses that the TSN CUC maintains.

        NOTE 2—If the MAC addresses used to construct
        Stream IDs are not unique within the network, duplicate
        Stream IDs can be generated, with unpredictable results.

      Unique ID:

        A 16-bit unique ID that is used to distinguish
        between multiple Streams within the station
        identified by MAC Address.

      The string specifies eight octets, with
      each octet represented as two hexadecimal
      characters. The first six octets specify
      the MAC Address, using the canonical format of
      IEEE Std 802, with a dash separating each octet.
      The last two octets specify the Unique ID,
      with the high-order octet, a dash, and then the
      low-order octet. The MAC Address and Unique ID
      are separated by colon.

      stream-id-type is intended for use by other modules
      as the type for a key to a list of Stream
      configurations (using group-talker and group-listener)
      and a list of Stream status (using group-status-stream
      and group-status-talker-listener).";
    reference
      "46.2.3.1 of IEEE Std 802.1Qcc-2018";
  }

  grouping group-interface-id {
    description
      "This YANG grouping specifies the identification of
      a distinct point of attachment (interface) in a station
      (end station or Bridge).";
    reference
      "46.2.3.3 of IEEE Std 802.1Qcc-2018";
    leaf mac-address {
      type string {
        pattern '[0-9a-fA-F]{2}(-[0-9a-fA-F]{2}){5}';
      }
      description
        "mac-address is the EUI-48 MAC address (IEEE Std 802)
        of the interface in the station (end station or Bridge).
        This MAC address uniquely identifies the station within
        the local network.

        mac-address shall be included in an instance of
        a container using group-interface-id.

        NOTE—This MAC address can be discovered in the
```

```
        physical topology using protocols such as
        IEEE Std 802.1AB (LLDP). LLDP supports MAC address
        as a subtype for the station's Chassis ID and Port ID.
        If the station does not use MAC address for its LLDP IDs,
        remote management can be used to associate this mac-address
        to the values provided in the LLDP IDs.

        The string uses the hexadecimal representation
        specified in IEEE Std 802 (i.e. canonical format).";
    }
    leaf interface-name {
      type string;
      description
        "interface-name is the name of the interface that is
        assigned locally by the station (end station or Bridge).

        interface-name may be included in an instance of
        a container using group-interface-id.

        IEEE Std 802 recommends that each distinct point of
        attachment to an IEEE 802 network have its own EUI MAC
        address. If the identified station follows this
        IEEE 802 recommendation, the mac-address leaf
        uniquely identifies the interface as well as the
        station, and interface-name is not needed.

        If the mac-address applies to more than one interface
        (distinct point of attachment) within the station,
        interface-name provides a locally assigned name that
        can help to identify the interface.

        When YANG is used for management of the station,
        interface-name is the interface name that serves
        as the key for the station's interface list (RFC7223).

        NOTE 1—The TSN CNC is typically located in a different
        physical product than the station identified by this
        group-interface-id. Since the interface-name is assigned
        locally by the identified station, it is possible that
        the station's product will change interface-name in a
        manner that the TSN CNC cannot detect. For example,
        RFC7223 mentions that the YANG interface name can change
        when a physical attachment point is inserted or removed.

        NOTE 2—This interface name can be discovered in the
        physical topology using protocols such as
        IEEE Std 802.1AB (LLDP). LLDP supports interface name
        as a subtype for its Port ID. If the station does not
        use interface name for its LLDP Port ID, remote management
        can be used to associate this interface-name to the values
        provided in the LLDP Port ID.";
    }
}
```

```
grouping group-ieee802-mac-addresses {
  description
    "This YANG grouping specifies the pair of
    IEEE 802 MAC addresses for Stream identification.

    The use of these fields for Stream identification
    corresponds to the managed objects for
    Stream identification in IEEE Std 802.1CB.
    If inconsistency arises between this specification
    and IEEE Std 802.1CB, IEEE Std 802.1CB takes
    precedence.";
  reference
    "46.2.3.4.1 of IEEE Std 802.1Qcc-2018";
  leaf destination-mac-address {
    type string {
      pattern '[0-9a-fA-F]{2}(-[0-9a-fA-F]{2}){5}';
    }
    description
      "Destination MAC address.

      An address of all 1's specifies that
      the destination MAC address is ignored for
      purposes of Stream identification.

      The string uses the hexadecimal representation
      specified in IEEE Std 802 (i.e. canonical format).";
  }
  leaf source-mac-address {
    type string {
      pattern '[0-9a-fA-F]{2}(-[0-9a-fA-F]{2}){5}';
    }
    description
      "Source MAC address.

      An address of all 1's specifies that
      the source MAC address is ignored for
      purposes of Stream identification.

      The string uses the hexadecimal representation
      specified in IEEE Std 802 (i.e. canonical format).";
  }
}

grouping group-ieee802-vlan-tag {
  description
    "This YANG grouping specifies a
    customer VLAN Tag (C-TAG of clause 9)
    for Stream identification.

    The Drop Eligible Indicator (DEI) field is
    not relevant from the perspective of a
    TSN Talker/Listener.

    The use of these fields for Stream identification
```

```
      corresponds to the managed objects for
      Stream identification in IEEE Std 802.1CB.
      If inconsistency arises between this specification
      and IEEE Std 802.1CB, IEEE Std 802.1CB takes
      precedence.";
    reference
      "46.2.3.4.2 of IEEE Std 802.1Qcc-2018";
    leaf priority-code-point {
      type uint8 {
        range "0 .. 7"; // 3 bits
      }
      description
        "Priority Code Point (PCP) field.

        The priority-code-point is not used to
        identify the Stream, but it does
        identify a traffic class (queue) in
        Bridges.";
    }
    leaf vlan-id {
      type uint16 {
        range "0 .. 4095"; // 12 bits
      }
      description
        "VLAN ID (VID) field.

        If only the priority-code-point is known,
        the vlan-id is specified as 0.";
    }
  }

  grouping group-ipv4-tuple {
    description
      "This YANG grouping specifies parameters
      to identify an IPv4 (RFC791) Stream.

      The use of these fields for Stream identification
      corresponds to the managed objects for
      Stream identification in IEEE Std 802.1CB.
      If inconsistency arises between this specification
      and IEEE Std 802.1CB, IEEE Std 802.1CB takes
      precedence.";
    reference
      "46.2.3.4.3 of IEEE Std 802.1Qcc-2018";
    leaf source-ip-address {
      type inet:ipv4-address;
      description
        "Source IPv4 address.

        An address of all 0's specifies that
        the IP source address is ignored for
        purposes of Stream identification.";
    }
    leaf destination-ip-address {
```

```
    type inet:ipv4-address;
    description
      "Destination IPv4 address.";
  }
  leaf dscp {
    type uint8;
    description
      "Differentiated services code
      point, DSCP (RFC2474).

      A value of 64 decimal specifies that
      the DSCP is ignored for
      purposes of Stream identification.";
  }
  leaf protocol {
    type uint16;
    description
      "IPv4 Protocol (e.g. UDP).

      The special value of all 1's (FFFF hex)
      represents 'None', meaning that
      protocol, source-port, and
      destination-port are ignored for
      purposes of Stream identification.

      For any value other than all 1's, the
      lower octet is used to match IPv4 Protocol.";
  }
  leaf source-port {
    type uint16;
    description
      "This matches the source port of the protocol.";
  }
  leaf destination-port {
    type uint16;
    description
      "This matches the destination port of the protocol.";
  }
}

grouping group-ipv6-tuple {
  description
    "This YANG grouping specifies parameters
    to identify an IPv6 (RFC2460) Stream.

    The use of these fields for Stream identification
    corresponds to the managed objects for
    Stream identification in IEEE Std 802.1CB.
    If inconsistency arises between this specification
    and IEEE Std 802.1CB, IEEE Std 802.1CB takes
    precedence.";
  reference
    "46.2.3.4.4 of IEEE Std 802.1Qcc-2018";
  leaf source-ip-address {
```

```
    type inet:ipv6-address;
    description
      "Source IPv6 address.

      An address of all 0's specifies that
      the IP source address is ignored for
      purposes of Stream identification.";
  }
  leaf destination-ip-address {
    type inet:ipv6-address;
    description
      "Destination IPv6 address.";
  }
  leaf dscp {
    type uint8;
    description
      "Differentiated services code
      point, DSCP (RFC2474).

      A value of 64 decimal specifies that
      the DSCP is ignored for
      purposes of Stream identification.";
  }
  leaf protocol {
    type uint16;
    description
      "IPv6 Next Header (e.g. UDP).

      The special value of all 1's (FFFF hex)
      represents 'None', meaning that
      protocol, source-port, and
      destination-port are ignored for
      purposes of Stream identification.

      For any value other than all 1's, the
      lower octet is used to match IPv6 Next Header.";
  }
  leaf source-port {
    type uint16;
    description
      "This matches the source port of the protocol.";
  }
  leaf destination-port {
    type uint16;
    description
      "This matches the destination port of the protocol.";
  }
}

grouping group-user-to-network-requirements {
  description
    "This YANG grouping specifies specifies user requirements
    for the Stream, such as latency and redundancy.
```

```
  The network (e.g. CNC) will merge
  all user-to-network-requirements for a Stream
  to ensure that all requirements are met.";
reference
  "46.2.3.6 of IEEE Std 802.1Qcc-2018";
leaf num-seamless-trees {
  type uint8;
  default "1";
  description
    "num-seamless-trees specifies the number
    of trees that the network will configure to
    deliver seamless redundancy for the Stream.

    The value zero is interpreted as one
    (i.e. no seamless redundancy).

    This requirement is provided from the Talker only.
    Listeners shall set this leaf to one.

    From each Talker to a single Listener, the
    network configures a path that relays Stream data
    through Bridges. If the Talker has more
    than one Listener, the network configures a
    tree of multiple paths.

    num-seamless-trees specifies the number of maximally
    disjoint trees that the network shall configure
    from the Talker to all Listeners. Each
    tree is disjoint from other trees, in that the
    network evaluates the physical topology to avoid
    sharing the same Bridge and links in each
    tree's paths. This computation of disjoint trees
    is maximal, in that shared Bridges and links
    are avoided to the maximum extent allowed
    by the physical topology. For example, if a
    single link exists from a Bridge to a Listener,
    and num-seamless-trees is 3, then all 3 trees will
    share that link to the Listener.

    When num-seamless-trees is greater than one,
    the transfer of the Stream's data frames
    shall use a seamless redundancy standard, such as
    IEEE Std 802.1CB. When a link shared by multiple trees
    diverges to multiple disjoint links, the
    seamless redundancy standard replicates
    (i.e. forwards a distinct copy of each data frame
    to the disjoint trees). When disjoint trees
    converge to a single link, the seamless redundancy
    standard eliminates the duplicate copies of each
    data frame. Assuming that other sources of frame loss
    are mitigated (e.g. congestion), failure of a link or
    Bridge in one disjoint tree does not result in frame
    loss as long as at least one remaining disjoint tree
    is operational.
```

```
      If the Talker sets this leaf to one, the network
      may make use of redundancy standards that are
      not seamless (i.e. failure of link results in
      lost frames), such as MSTP and IS-IS.

      If the Talker sets this leaf to greater than one,
      and seamless redundancy is not possible in the
      current network (no disjoint paths, or no seamless
      redundancy standard in Bridges),
      group-status-stream.status-info.failure-code
      is non-zero (46.2.4.1 of IEEE Std 802.1Qcc-2018).

      If group-user-to-network-requirements is not
      provided by the Talker or Listener, the network
      shall use the default value of one for this leaf.";
    reference
      "46.2.3.6.1 of IEEE Std 802.1Qcc-2018";
  }
  leaf max-latency {
    type uint32;
    default "0";
    description
      "Maximum latency from Talker to
      Listener(s) for a single frame of the Stream.

      max-latency is specified as an integer number
      of nanoseconds.

      Latency shall use the definition of 3.102,
      with additional context as follows:
      The 'Known reference point in the frame' is
      the message timestamp point specified in
      IEEE Std 802.1AS for various media
      (i.e. start of the frame). The 'first point'
      is in the Talker, at the reference plane
      marking the boundary between the network
      media and PHY (see IEEE Std 802.1AS).
      The 'second point' is in the Listener,
      at the reference plane marking the boundary
      between the network media and PHY.

      When this requirement is specified by
      the Talker, it must be satisfied for
      all Listeners.

      When this requirement is specified by
      the Listener, it must be satisfied for
      this Listener only.

      If group-user-to-network-requirements is
      not provided by the Talker or Listener,
      the network shall use the default value of
      zero for this leaf.
```

```
        The special value of zero represents
        usage of the initial value of
        group-status-talker-listener.accumulated-latency
        as the maximum latency requirement. This effectively
        locks-down the initial latency that the network
        calculates after successful configuration of the
        Stream, such that any subsequent increase in
        latency beyond that value causes the Stream to fail.

        The assumption for when the 'first point' occurs
        in the Talker depends on the presence of the
        time-aware container in the Talker's
        traffic-specification.

        When time-aware is not present:

          The Talker is assumed to transmit
          at an arbitrary time (not scheduled).

        When time-aware is present:

          The 'first point' is assumed to occur
          at the start of traffic-specification.interval,
          as if the Talker's offsets (earliest-transmit-offset
          and latest-transmit-offset) are both zero.
          The Talker's offsets are not typically zero,
          but use of the start of interval for purposes
          of max-latency allows the Listener(s) to
          schedule their application independently
          from the Talker's offset configuration.

          The Listener determines max-latency
          based on its scheduling of a read
          function in the application. Nevertheless,
          the time from frame reception (i.e. 'second
          point') to execution of the read function
          is in the user scope, and therefore
          not included in max-latency.

          max-latency can be set to
          a value greater than the Talker's
          interval, in order to specify a
          longer latency requirement. For example,
          if the Talker's interval is 500 microsec,
          and max-latency is 700 microsec, the Listener
          receives the frame no later than
          200 microsec into the interval that follows
          the Talker's interval.";
      reference
        "46.2.3.6.2 of IEEE Std 802.1Qcc-2018";
    }
}
```

```
grouping group-interface-capabilities {
  description
    "This YANG grouping specifies the network
    capabilities of all interfaces (Ports) contained
    in end-station-interfaces.

    The network may provide configuration
    of these capabilities in
    group-status-talker-listener.interface-configuration.

    NOTE—If an end station contains multiple interfaces
    with different network capabilities, each interface
    should be specified as a distinct Talker or
    Listener (i.e. one entry in end-station-interfaces).
    Use of multiple entries in end-station-interfaces is intended
    for network capabilities that span multiple interfaces
    (e.g. seamless redundancy).";
  reference
    "46.2.3.7 of IEEE Std 802.1Qcc-2018";
  leaf vlan-tag-capable {
    type boolean;
    default "false";
    description
      "When vlan-tag-capable is true, the interface supports
      the ability to tag/untag frames using a
      Customer VLAN Tag (C-TAG of clause 9)
      provided by the network.

      For a Talker, the network's tag replaces the
      tag specified by the data-frame-specification.
      If the data-frame-specification is untagged
      (no group-ieee802-vlan-tag), the network's tag is
      inserted in the frame as it passes through the
      interface.

      For a Listener, the user's tag from
      the data-frame-specification replaces the
      network's tag as the frame passes
      through the interface. If the data-frame-specification
      is untagged (no group-ieee802-vlan-tag), the
      network's tag is removed from the frame as it
      passes through the interface.

      If the end station supports more than one interface
      (i.e. more than one entry in end-station-interfaces),
      vlan-tag-capable of true means that a distinct
      VLAN tag can be applied to each interface. The list
      of VLAN tag (one for each interface) can be provided
      by the network in interface-configuration.interface-list
      (ieee802-vlan-tag choice).

      When vlan-tag-capable is false, the interface
      does not support the capability to tag/untag frames
      using a Customer VLAN Tag (C-TAG of clause 9)
```

```
      provided by the network.

      If interface-capabilities is not provided by the Talker
      or Listener, the network shall use the default
      value of false for this leaf.";
    reference
      "46.2.3.7.1 of IEEE Std 802.1Qcc-2018";
  }
  leaf-list cb-stream-iden-type-list {
    type uint32;
    description
      "cb-stream-iden-type-list provides a list of the
      supported Stream Identification types as specified
      in IEEE Std 802.1CB.

      Each Stream Identification type is provided as a
      32-bit unsigned integer. The upper three octets
      contain the OUI/CID, and the lowest octet contains
      the type number.

      NOTE—If the Talker/Listener end system supports
      IEEE Std 802.1CB, Null Stream identification is
      required, and that Stream Identification type is
      included in this list. If the Talker/Listener end
      system does not support IEEE Std 802.1CB, this
      list is empty.

      If the end station supports more than one interface
      (i.e. more than one interface-id in end-station-interfaces,
      an empty cb-stream-iden-type-list means that the end station
      is capable of transferring the Stream on any one of its
      interfaces (not all). When this is specified, the network
      shall decide which interface is best used for TSN purposes,
      and communicate that decision by returning a single interface
      in interface-configuration.interface-list. The
      Talker/Listener uses this interface alone for the Stream.

      If interface-capabilities is not provided within
      group-talker or group-listener, the network shall use an empty
      list as the default value for this element.";
    reference
      "46.2.3.7.2 of IEEE Std 802.1Qcc-2018";
  }
  leaf-list cb-sequence-type-list {
    type uint32;
    description
      "cb-sequence-type-list provides a list of the supported
      Sequence Encode/Decode types as specified in
      IEEE Std 802.1CB.

      Each sequence type is provided as a 32-bit unsigned
      integer. The upper three octets contain the OUI/CID,
      and the lowest octet contains the type number.
```

```
      If interface-capabilities is not provided within
      group-talker or group-listener, the network shall use an empty
      list as the default value for this element.";
    reference
      "46.2.3.7.3 of IEEE Std 802.1Qcc-2018";
  }
}

grouping group-interface-configuration {
  description
    "This YANG grouping provides configuration of
    interfaces in the Talker/Listener. This configuration
    assists the network in meeting the Stream's requirements.
    The interface-configuration meets the capabilities of
    the interface as provided in interface-capabilities.";
  reference
    "46.2.5.3 of IEEE Std 802.1Qcc-2018";
  list interface-list {
    key "mac-address interface-name";
    description
      "A distinct configuration is provided for
      each interface in the Talker/Listener (even if
      multiple interfaces use the same configuration).
      Each entry in this interface-list consists
      of an interface identification (group-interface-id),
      followed by a list of configuration values for
      that interface (config-list).

      If interface-configuration is not provided within
      group-status-talker-listener, the network shall
      assume zero entries as the default (no interface
      configuration).

      Since the interface-name leaf is optional, empty string
      can be used for its key value.";
    uses group-interface-id;
    list config-list {
      key "index";
      description
        "List of configuration values for
        the interface.";
      leaf index {
        type uint8;
        description
          "This index is provided in order to
          provide a unique key per list entry.
          The value of index for each entry
          shall be unique (but not necessarily
          contiguous).";
      }
      choice config-value {
        description
          "One of the following choices is
          provided for each configuration value.
```

```
      Each container name acts as the case name
      for the choice.";
    container ieee802-mac-addresses {
      description
        "Source and destination MAC addresses
        that apply to the network side of
        the user/network boundary.

        NOTE 1—On the userside, the MAC addresses
        correspond to the ieee802-mac-addresses
        of data-frame-specification.

        NOTE 2—The source MAC address of the
        network is typically the same as the
        user. The destination MAC address can
        be different. For example, the user
        can use an individual address, but
        the network can use a group (multicast)
        address.

        This configuration value is not provided
        unless IEEE Std 802.1CB is supported, and
        a value for Active Destination MAC
        and VLAN Stream identification
        is provided in cb-stream-iden-type-list
        of interface-capabilities.";
      reference
        "46.2.5.3.1 of IEEE Std 802.1Qcc-2018";
      uses group-ieee802-mac-addresses;
    }
    container ieee802-vlan-tag {
      description
        "Customer VLAN Tag (C-TAG of clause 9)
        that applies to the network side of
        the user/network boundary.

        NOTE—On the user side, the VLAN tag corresponds
        to the ieee802-vlan-tag of data-frame-specification
        (including untagged if this field is not provided).

        If the user provides a VLAN ID in the
        ieee802-vlan-tag of data-frame-specification,
        the Stream's data frames are assumed to
        be limited to the active topology for
        that VLAN ID. Therefore, if the network
        uses a different VLAN ID in
        this config-value, the network shall ensure
        that the replacement VLAN ID is limited
        to the equivalent active topology.

        This configuration value is not provided
        unless vlan-tag-capable of
        interface-capabilities is true.";
      reference
```

```
        "46.2.5.3.2 of IEEE Std 802.1Qcc-2018";
      uses group-ieee802-vlan-tag;
    }
    container ipv4-tuple {
      description
        "IPv4 identification that applies to the
        network side of the user/network
        boundary.

        This configuration value is not provided
        unless IEEE Std 802.1CB is supported,
        and a value for IP Stream identification
        is provided in cb-stream-iden-type-list
        of interface-capabilities.";
      reference
        "46.2.5.3.3 of IEEE Std 802.1Qcc-2018";
      uses group-ipv4-tuple;
    }
    container ipv6-tuple {
      description
        "IPv6 identification that applies to the
        network side of the user/network
        boundary.

        This configuration value is not provided
        unless IEEE Std 802.1CB is supported,
        and a value for IP Stream identification
        is provided in cb-stream-iden-type-list
        of interface-capabilities.";
      reference
        "46.2.5.3.4 of IEEE Std 802.1Qcc-2018";
      uses group-ipv6-tuple;
    }
    leaf time-aware-offset {
      type uint32;
      description
      "If the time-aware container
      is present in the
      traffic-specification of the Talker,
      this config-value shall be provided
      by the network to the Talker.

      If the time-aware container
      is not present in the
      traffic-specification of the Talker,
      this config-value shall not
      be provided by the network.

      This config-value shall not
      be provided to Listeners, as it is
      not applicable.

      time-aware-offset specifies
      the offset that the Talker
```

```
            shall use for transmit.
            The network returns a value between
            earliest-transmit-offset
            and latest-transmit-offset of the
            Talker's traffic-specification.
            The value is expressed as
            nanoseconds after the start
            of the Talker's interval.";
            reference
              "46.2.5.3.5 of IEEE Std 802.1Qcc-2018";
          }
        }
      }
    }
  }

  grouping group-talker {
    description
      "This YANG grouping specifies:
      - Talker's behavior for Stream (how/when transmitted)
      - Talker's requirements from the network
      - TSN capabilities of the Talker's interface(s)

      In the fully centralized model of TSN configuration,
      this grouping originates from the CUC, and
      is delivered to the CNC.";
    reference
      "46.2.3 of IEEE Std 802.1Qcc-2018";

    container stream-rank {
      description
        "Rank of this Stream's configuration relative to other
        Streams in the network. This rank is used to determine
        success/failure of Stream resource configuration,
        and it is unrelated to the Stream's data.";
      reference
        "46.2.3.2 of IEEE Std 802.1Qcc-2018";
      leaf rank {
        type uint8;
        description
          "The Rank is used by the network to decide which Streams
          can and cannot exist when TSN resources reach their limit.
          If a Bridge's Port becomes oversubscribed (e.g. network
          reconfiguration, IEEE 802.11 bandwidth reduction), the
          Rank is used to help determine which Streams can be
          dropped (i.e. removed from Bridge configuration).

          The only valid values for Rank shall be zero and one.
          The configuration of a Stream with Rank zero is more
          important than the configuration of a Stream with
          Rank one. The Rank value of zero is intended for
          emergency traffic, and the Rank value of one is
          intended for non-emergency traffic.
```

```
        NOTE—It is expected that higher layer applications
        and protocols can use the Rank to indicate the
        relative importance of Streams based on user
        preferences. Those user preferences are expressed
        by means beyond the scope of this standard. When
        multiple applications exist in a network
        (e.g. audio/video along with industrial control),
        it can be challenging for the varied applications
        and vendors to agree on multiple Rank values.
        To mitigate such challenges, this Rank uses
        a simple concept of emergency (zero) and
        non-emergency (one) that can be applied
        over all applications. For example, in a network
        that carries audio Streams for fire safety
        announcements, all applications are likely to
        agree that those Streams use Rank of zero.";
      reference
        "46.2.3.2.1 of IEEE Std 802.1Qcc-2018";
    }
  }

  list end-station-interfaces {
    key "mac-address interface-name";
    min-elements 1;
    description
      "List of identifiers, one for each physical
      interface (distinct point of attachment) in
      the end station acting as a Talker.

      Although many end stations contain a single interface,
      this list allows for multiple interfaces. Some TSN
      features allow a single Stream to span multiple interfaces
      (e.g. seamless redundancy).

      Each entry of end-station-interfaces is used by the CNC
      to locate the Talker in the topology.

      Since the interface-name leaf is optional, empty string
      can be used for its key value.";
    reference
      "46.2.3.3 of IEEE Std 802.1Qcc-2018";
    uses group-interface-id;
  }

  list data-frame-specification {
    key "index";
    min-elements 1;
    description
      "data-frame-specification specifies the frame that carries the
      Talker's Stream data. The network uses the specification
      to identify this Stream's frames as TSN, in order to apply
      the required TSN configuration.

      The specification is based on the user's knowledge of the
```

```
      frame, without any network specifics. In other words, this
      specifies the frame that the Talker would use in the absence
      of TSN.

      The specification is provided as a list of fields
      that the user knows. The list is ordered
      from start of frame to end of header.
      For example, if the Talker uses a VLAN-tagged
      Ethernet frame (not IP), the list consists of
      ieee802-mac-addresses followed by ieee802-vlan-tag.
      For example, if the Talker uses a UDP/IPv4 packet
      without knowledge of the Ethernet header,
      the list consists of ipv4-tuple.

      This list is optional, and its absence
      indicates that Stream transformation is performed
      in the Talker and Listeners of this Stream
      (46.2.2 of IEEE Sd 802.1Q-2018).";
    reference
      "46.2.3.4 of IEEE Std 802.1Qcc-2018";
    leaf index {
      type uint8;
      description
        "This index is provided in order to
        provide a unique key per list entry.
        The value of index for each entry
        shall be unique (but not necessarily
        contiguous).";
    }
    choice field {
      description
        "One of the following choices is provided
        for each field that the user knows.
        Each container name acts as the case name
        for the choice.";
      container ieee802-mac-addresses {
        description "IEEE 802 MAC addresses.";
        uses group-ieee802-mac-addresses;
      }
      container ieee802-vlan-tag {
        description "IEEE 802.1 CTAG";
        uses group-ieee802-vlan-tag;
      }
      container ipv4-tuple {
        description "IPv4 packet identification";
        uses group-ipv4-tuple;
      }
      container ipv6-tuple {
        description "IPv6 packet identification";
        uses group-ipv6-tuple;
      }
    }
  }
```

```
container traffic-specification {
  description
    "This traffic-specification specifies how the Talker
    transmits frames for the Stream. This is effectively
    the Talker's promise to the network. The network
    uses this traffic spec to allocate resources and
    adjust queue parameters in Bridges.";
  reference
    "46.2.3.5 of IEEE Std 802.1Qcc-2018";
  container interval {
    description
      "This interval specifies the period of time in
      which the traffic specification cannot be exceeded.
      The traffic specification is specified by
      max-frames-per-interval and max-frame-size.

      The interval is a rational number of seconds,
      defined by an integer numerator and an integer
      denominator.

      If the time-aware container is not present,
      the interval specifies a sliding window of time.
      The Talker's transmission is not synchronized
      to a time on the network, and therefore
      the traffic specification cannot be exceeded
      during any interval in time.

      If the time-aware container is present,
      the interval specifies a window of time that is
      aligned with the time epoch that is synchronized
      on the network. For example, if IEEE Std
      802.1AS-2011 is used with the PTP timescale,
      the first interval begins at 1 January 00:00:00 TAI.
      If CurrentTime represents the current time, then
      the start of the next interval (StartOfNextInterval)
      is:
        StartOfNextInterval = N * interval
      where N is the smallest integer for which the relation
        StartOfNextInterval >= CurrentTime
      would be TRUE.";
    reference
      "46.2.3.5.1 of IEEE Std 802.1Qcc-2018";
    leaf numerator {
      type uint32;
      description "interval's numerator.";
    }
    leaf denominator {
      type uint32;
      description "interval's denominator.";
    }
  }
  leaf max-frames-per-interval {
    type uint16;
    description
```

```
    "max-frames-per-interval specifies the maximum
    number of frames that the Talker can transmit
    in one interval.";
  reference
    "46.2.3.5.2 of IEEE Std 802.1Qcc-2018";
}
leaf max-frame-size {
  type uint16;
  description
    "max-frame-size specifies maximum frame size that
    the Talker will transmit, excluding any overhead
    for media-specific framing (e.g., preamble,
    IEEE 802.3 header, Priority/VID tag, CRC,
    interframe gap). As the Talker or Bridge determines
    the amount of bandwidth to reserve on the
    egress Port (interface), it will calculate the
    media-specific framing overhead on that Port and
    add it to the number specified in the max-frame-size
    leaf.";
  reference
    "46.2.3.5.3 of IEEE Std 802.1Qcc-2018";
}
leaf transmission-selection {
  type uint8;
  description
    "transmission-selection specifies the algorithm
    that the Talker uses to transmit this Stream's
    traffic class. This algorithm is often referred
    to as the shaper for the traffic class.

    The value for this leaf uses Table 8-5
    (Transmission selection algorithm identifiers)
    of 8.6.8 of IEEE Std 802.1Q-2018.
    If no algorithm is known, the value
    zero (strict priority) can be used.

    The Talker's shaping and scheduling of the
    Stream is considered to be on the user side
    of the user/network boundary, and this leaf
    specifies the Talker's behavior to the network.";
  reference
    "46.2.3.5.4 of IEEE Std 802.1Qcc-2018";
}
container time-aware {
  presence
    "Specifies that the Talker's traffic is synchronized
    to a known time on the network
    (e.g. using IEEE Std 802.1AS)";
  description
    "The time-aware container provides leafs to specify
    the Talker's time-aware transmit to the network.

    The Talker and Listeners of a Stream are assumed to
    coordinate using user (application) mechanisms, such
```

```
    that each Listener is aware that its Talker transmits
    in a time-aware manner.

    If max-frames-per-interval is greater than one,
    the Talker shall transmit multiple frames as a burst
    within the interval, with the minimum inter-frame gap
    allowed by the media.

    NOTE—Although scheduled traffic (8.6.8.4 of
    IEEE Std 802.1Q-2018) specifies a valid implementation
    of a time-aware Talker, the time-aware container
    is intended to support alternate implementations of
    scheduling.";
  reference
    "46.2.3.5 of IEEE Std 802.1Qcc-2018";
  leaf earliest-transmit-offset {
    type uint32;
    description
      "earliest-transmit-offset specifies the
      earliest offset within the interval at which
      the Talker is capable of starting
      transmit of its frames. As part of
      group-status-talker-listener.interface-configuration,
      the network will return a specific
      time-aware-offset to the Talker
      (within the earliest/latest range),
      which the Talker uses to schedule its transmit.

      earliest-transmit-offset is specified
      as an integer number of nanoseconds.

      The Talker's transmit offsets
      include earliest-transmit-offset,
      latest-transmit-offset, and the
      time-aware-offset returned to the Talker.
      Each of the Talker's offsets is specified
      at the point when the message timestamp point
      of the first frame of the Stream passes the
      reference plane marking the boundary between
      the network media and PHY.
      The message timestamp point is specified
      by IEEE Std 802.1AS for various media.";
    reference
      "46.2.3.5.5 of IEEE Std 802.1Qcc-2018";
  }
  leaf latest-transmit-offset {
    type uint32;
    description
      "latest-transmit-offset specifies the
      latest offset within the interval at which
      the Talker is capable of starting
      transmit ofits frames. As part of
      group-status-talker-listener.interface-configuration,
      the network will return a specific
```

```
        time-aware-offset to the Talker
        within the earliest/latest range),
        which the Talker uses to schedule its transmit.

        latest-transmit-offset is specified
        as an integer number of nanoseconds.";
      reference
        "46.2.3.5.6 of IEEE Std 802.1Qcc-2018";
    }
    leaf jitter {
      type uint32;
      description
        "The jitter leaf specifies the maximum difference
        in time between the Talker's transmit offsets,
        and the ideal synchronized network time
        (e.g. IEEE 802.1AS time). Jitter is
        specified as an unsigned integer number
        of nanoseconds.

        The maximum difference means
        sooner or later than the ideal (e.g. transmit
        +/- 500 nanoseconds relative to IEEE 802.1AS time
        results in jitter of 500).

        The ideal synchronized network time refers to
        time at the source (e.g. IEEE 802.1AS grandmaster).
        The jitter does not include inaccuracies as
        time is propagated from the time source to the
        Talker, because those inaccuracies are
        assumed to be known by the network, and
        time synchronization is a network technology.
        The jitter leaf is intended to specify
        inaccuracies in the Talker's implementation.
        For example, if the Talker's IEEE 802.1AS time is
        +/- 812 nanoseconds relative to the
        grandmaster, and the Talker schedules using a
        100 microsecond timer tick driven by IEEE 802.1AS
        time, Jitter is 50000 (not 50812).

        The Talker's transmit offsets
        include earliest-transmit-offset,
        latest-transmit-offset, and the
        time-aware-offset returned to the Talker in
        group-status-talker-listener.interface-configuration.";
      reference
        "46.2.3.5.7 of IEEE Std 802.1Qcc-2018";
    }
  }
}

container user-to-network-requirements {
  description
    "user-to-network-requirements specifies user requirements
    for the Stream, such as latency and redundancy.
```

```
      The network (CNC) will merge all
      user-to-network-requirements for a Stream
      to ensure that all requirements are met.";
    reference
      "46.2.3.6 of IEEE Std 802.1Qcc-2018";
    uses group-user-to-network-requirements;
  }

  container interface-capabilities {
    description
      "interface-capabilities specifies the network
      capabilities of all interfaces (Ports) contained
      in end-station-interfaces.

      The network may provide configuration
      of these capabilities in
      group-status-talker-listener.interface-configuration.

      NOTE—If an end station contains multiple interfaces
      with different network capabilities, each interface
      should be specified as a distinct Talker or
      Listener (i.e. one entry in end-station-interfaces).
      Use of multiple entries in end-station-interfaces is intended
      for network capabilities that span multiple interfaces
      (e.g. seamless redundancy).";
    reference
      "46.2.3.7 of IEEE Std 802.1Qcc-2018";
    uses group-interface-capabilities;
  }
}

grouping group-listener {
  description
    "This YANG grouping specifies:
    - Listener's requirements from the network
    - TSN capabilities of the Listener's interface(s)

    In the fully centralized model of TSN configuration,
    this grouping originates from the CUC, and
    is delivered to the CNC.";
  reference
    "46.2.4 of IEEE Std 802.1Qcc-2018";

  list end-station-interfaces {
    key "mac-address interface-name";
    min-elements 1;
    description
      "List of identifiers, one for each physical
      interface (distinct point of attachment) in
      the end station acting as a Listener.

      Although many end stations contain a single interface,
      this list allows for multiple interfaces. Some TSN
      features allow a single Stream to span multiple interfaces
```

```
        (e.g. seamless redundancy).

        Each entry of end-station-interfaces is used by the CNC
        to locate the Listener in the topology.

        Since the interface-name leaf is optional, empty string
        can be used for its key value.";
      reference
        "46.2.3.3 of IEEE Std 802.1Qcc-2018";
      uses group-interface-id;
    }

    container user-to-network-requirements {
      description
        "user-to-network-requirements specifies user requirements
        for the Stream, such as latency and redundancy.
        The network (CNC) will merge all
        user-to-network-requirements for a Stream
        to ensure that all requirements are met.";
      reference
        "46.2.3.6 of IEEE Std 802.1Qcc-2018";
      uses group-user-to-network-requirements;
    }

    container interface-capabilities {
      description
        "interface-capabilities specifies the network
        capabilities of all interfaces (Ports) contained
        in end-station-interfaces.

        The network may provide configuration
        of these capabilities in
        group-status-talker-listener.interface-configuration.

        NOTE—If an end station contains multiple interfaces
        with different network capabilities, each interface
        should be specified as a distinct Talker or
        Listener (i.e. one entry in end-station-interfaces).
        Use of multiple entries in end-station-interfaces is intended
        for network capabilities that span multiple interfaces
        (e.g. seamless redundancy).";
      reference
        "46.2.3.7 of IEEE Std 802.1Qcc-2018";
      uses group-interface-capabilities;
    }
  }

  grouping group-status-stream {
    description
      "This YANG grouping provides the status of a Stream's
      configuration from the network to each user. The status
      in this grouping applies to the entire Stream (Talker
      and all Listeners).
```

```
  In the fully centralized model of TSN configuration,
  this grouping originates from the CNC, and
  is delivered to the CUC.

  The group-status-stream and group-status-talker-listener
  groupings are intended to be used by other modules
  within a list of status (state) for each Stream,
  with each list entry using:
  - leaf of type stream-id-type, used as key to the list
  - container using group-status-stream
  - container for Talker, using group-status-talker-listener
  - list for Listeners, using group-status-talker-listener";
reference
  "46.2.5 of IEEE Std 802.1Qcc-2018";

container status-info {
  description
    "status-info provides information regarding the status
    of a Stream's configuration in the network.";
  reference
    "46.2.5.1 of IEEE Std 802.1Qcc-2018";
  leaf talker-status {
    type enumeration {
      enum none {
        value 0;
        description "No Talker detected.";
      }
      enum ready {
        value 1;
        description "Talker ready (configured).";
      }
      enum failed {
        value 2;
        description "Talker failed.";
      }
    }
    description
      "This is an enumeration for the status of
      the Stream's Talker.";
    reference
      "46.2.5.1.1 of IEEE Std 802.1Qcc-2018";
  }
  leaf listener-status {
    type enumeration {
      enum none {
        value 0;
        description "No Listener detected.";
      }
      enum ready {
        value 1;
        description "All Listeners ready (configured).";
      }
      enum partial-failed {
        value 2;
```

```
            description
              "One or more Listeners ready, and
              one or more Listeners failed.
              If Talker is ready, Stream can be used.";
          }
          enum failed {
            value 3;
            description "All Listeners failed";
          }
        }
        description
          "This is an enumeration for the status of
          the Stream's Listener(s).";
        reference
          "46.2.5.1.2 of IEEE Std 802.1Qcc-2018";
      }
      leaf failure-code {
        type uint8;
        description
          "If the Stream encounters a failure (talker-status
          is failed, or listener-status is failed, or
          listener-status is partial-failed), failure-code
          provides a non-zero code that specifies the
          problem. Table 46-1 of IEEE Std 802.1Q-2018
          describes each code.)";
        reference
          "46.2.5.1.3 of IEEE Std 802.1Qcc-2018";
      }
    }

    list failed-interfaces {
      key "mac-address interface-name";
      description
        "When a failure occurs in network configuration
        (i.e. non-zero failure-code in status-info),
        failed-interfaces provides a list of one or more
        physical interfaces (distinct points of attachement)
        in the failed end station or Bridge. Each identifier
        is sufficient to locate the interface in the physical
        topology.

        The failed-interfaces list is optional.

        Since the interface-name leaf is optional, empty string
        can be used for its key value.";
      reference
        "46.2.5.4 of IEEE Std 802.1Qcc-2018";
      uses group-interface-id;
    }
  }

  grouping group-status-talker-listener {
    description
      "This YANG grouping provides the status for a specific
```

```
    Talker or Listener.

    In the fully centralized model of TSN configuration,
    this grouping originates from the CNC, and
    is delivered to the CUC.";
  reference
    "46.2.5 of IEEE Std 802.1Qcc-2018";

  leaf accumulated-latency {
    type uint32;
    description
      "accumulated-latency provides the worst-case maximum
      latency that a single frame of the Stream
      can encounter along its current path(s).

      When provided to a Listener, accumulated-latency is the
      worst-case maximum latency for that Listener only.

      When provided to a Talker, accumulated-latency is the
      worst-case maximum latency for all Listeners (worst path).

      accumulated-latency is specified as an integer number
      of nanoseconds.

      accumulated-latency uses the same definition
      for latency as user-to-network-requirements.max-latency.

      For successful status-info, the network
      returns a value less than or equal to
      user-to-network-requirements.max-latency.

      If the time-aware container is present in
      the traffic-specification of the Talker,
      the value is expressed as nanoseconds after the
      start of the Talker's traffic-specification.interval.

      If the time-aware container is not present in
      the traffic-specification of the Talker,
      the value is expressed as nanoseconds after the
      Talker's transmit of any frame in the Stream,
      at any arbitrary time.

      If user-to-network-requirements.num-seamless-trees is one,
      accumulated-latency shall provide the worst-case maximum
      latency for the current path from Talker to each Listener.
      If the path is changed (e.g. by a spanning tree protocol),
      accumulated-latency changes accordingly.

      If user-to-network-requirements.num-seamless-trees
      is greater than one, accumulated-latency shall
      provide the worst-case maximum latency for all paths
      configured from the Talker to each Listener.";
    reference
      "46.2.5.2 of IEEE Std 802.1Qcc-2018";
```

```
    }

    container interface-configuration {
      description
        "interface-configuration provides configuration of
        interfaces in the Talker/Listener. This configuration
        assists the network in meeting the Stream's requirements.
        The interface-configuration meets the capabilities of
        the interface as provided in interface-capabilities.";
      reference
        "46.2.5.3 of IEEE Std 802.1Qcc-2018";

      uses group-interface-configuration;
    }
  }
}
```

Annex A

(normative)

PICS proforma—Bridge implementations[4]

A.5 Major capabilities

Change the following rows of the table in A.5 as shown:

Item	Feature	Status	References	Support
MGT	Is management of the Bridge supported?	O PBBTE OR TPMR OR SRRM: M	Clause 5, A.14	Yes [ ] No [ ]
RMGT	Is a remote management protocol supported?	MGT:O PBBTE OR TPMR OR SRRM: M	Clause 5, A.15	Yes [ ] No [ ]

Insert the following rows at the end of the table in A.5:

Item	Feature	Status	References	Support
SRRM	Does the implementation support Stream reservation remote management?	O	5.4.1.5, 12.32, A.49	Yes [ ] No [ ]
CNC-S	Does the implementation support the functionality of a Centralized Network Configuration (CNC) station?	O	5.29, 46.2, A.50	Yes [ ] No [ ]

A.31 Stream Reservation Protocol

Change the following rows of the table in A.31 as shown:

Item	Feature	Status	Reference	Support
SRP-1	Does the implementation support the exchange of Multiple Stream Registration Protocol Data Units (MSRPDUs), using the generic Multiple Registration Protocol Data Unit (MRPDU) format defined in 10.8 to exchange MSRP-specific information, as defined in 35.2.2.8.1, 35.2.2.9.1, and 35.2.2.10.1?	M	10.8, 35.2.2.8.1, 35.2.2.9.1, 35.2.2.10.1	Yes [ ]

[4]*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.

Item	Feature	Status	Reference	Support
SRP-7	Does the ProtocolVersion used for the implementation of MSRP take the hexadecimal value ~~0x00~~ 0x01?	M	35.2.2.3	Yes [ ]
SRP-11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 35.2.2.8, 35.2.2.9, and 35.2.2.10?	M	35.2.2.8, 35.2.2.9, 35.2.2.10	Yes [ ]
SRP-12	Does the implementation update Accumulated Latency as the Talker attributes propagate through the Bridge?	M	35.2.2.8.6, 35.2.2.10.10	Yes [ ]
SRP-13	Does the implementation update ~~the F~~failure ~~I~~information ~~Bridge ID and Code~~ in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.2.8.7, 35.2.2.10.9, 35.2.2.10.12	Yes [ ]
SRP-15	Is talkerPruning and MMRP supported?	O	~~35.2.4.3.1~~ 35.2.4.3.2	Yes [ ] No [ ]

Insert the following rows into the table in A.31 in numeric order:

Item	Feature	Status	Reference	Support
SRP-25	Does the implementation support both original and enhanced attribute types?	M	35.1	Yes [ ]
SRP-26	Does the implementation detect domain boundaries?	M	35.2.4.3	Yes [ ]
SRP-27	Does the implementation support Stream transformation in Bridge?	O	35.2.2.10.5	Yes [ ] No [ ]
SRP-28	Does the implementation support protocol version translation for Talker attributes?	M	35.2.4.3.1	Yes [ ]
SRP-29	Does the implementation support protocol version translation for Listener attributes?	M	35.2.4.4.4	Yes [ ]
SRP-30	Does the implementation declare the Domain attribute prior to Talker/ Listener attributes?	M	35.2.2.9	Yes [ ]
SRP-31	Is talker pruning per Port supported?	O	35.2.4.3.3	Yes [ ] No [ ]
SRP-32	Is talker VLAN pruning supported?	O	35.2.4.3.4	Yes [ ] No [ ]

Insert the following subclauses (A.49 and A.50) after A.48:

A.49 Stream reservation remote management (SRRM)

Item	Feature	Status	References	Support
	If Stream reservation remote management functionality (SRMM of A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A [ ]
SRRM-1	What management protocol standard(s) or specification(s) are supported (server side)?	M	5.4.1.10(b)	Yes [ ]
SRRM-2	Does the implementation report delay through the Bridge?	M	12.32.1	Yes [ ]
SRRM-3	Does the implementation report propagation delay?	M	12.32.2	Yes [ ]
SRRM-4	Does the implementation support Static Trees for static configuration of spanning trees?	M	5.4.1.10 item c), 12.32.3	Yes [ ]
SRRM-5	Does the implementation support MRP External Control for the MSRP application?	O SRP: M	12.32.4	Yes [ ] No [ ]
SRRM-6	Does the implementation support MRP External Control for the MVRP application?	O MVRP: M	12.32.4	Yes [ ] No [ ]
SRRM-7	Does the implementation support MRP External Control for the MMRP application?	O MMRP: M	12.32.4	Yes [ ] No [ ]
SRRM-8	Does the implementation support queue reservation for traffic classes using the strict priority algorithm, through configuration of adminIdleSlope?	M	5.4.1.10 item e), 12.20.1, 34.3	Yes [ ]

A.50 TSN Centralized Network Configuration (CNC) station

Item	Feature	Status	References	Support
	If the functionality of a Centralized Network Configuration station (CNC-S of A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A [ ]
CNC-S-1	What management protocol standard(s) or specification(s) are supported (client side)?	M	5.29 item a)	_____ _____ _____
CNC-S-2	Does the implementation support the managed object definitions and encodings for Stream reservation remote management?	M	5.29 item b), 12.32	Yes [ ]
CNC-S-3	Does the implementation support the managed object definitions and encodings for scheduled traffic?	O	12.29	Yes [ ] No [ ]

A.50 TSN Centralized Network Configuration (CNC) station *(continued)*

Item	Feature	Status	References	Support
CNC-S-4	Does the implementation support the managed object definitions and encodings for frame preemption?	O	12.30	Yes [ ] No [ ]
CNC-S-5	Does the implementation support the managed object definitions and encodings for IEEE Std 802.1AS?	O	IEEE Std 802.1AS	Yes [ ] No [ ]
CNC-S-6	Does the implementation support the managed object definitions and encodings for IEEE Std 802.1CB?	O	IEEE Std 802.1CB	Yes [ ] No [ ]
CNC-S-7	Does the implementation support MRP External Control for the MSRP application?	O	12.32.4	Yes [ ] No [ ]
CNC-S-8	Does the implementation support MRP External Control for the MVRP application?	O	12.32.4	Yes [ ] No [ ]
CNC-S-9	Does the implementation support MRP External Control for the MMRP application?	O	12.32.4	Yes [ ] No [ ]
CNC-S-10	What user/network configuration protocol standard(s) or specification(s) are supported?	M	5.29 item c), 46.2.2	_____ _____ _____
CNC-S-11	Does the implementation conform to the conditional requirements for use of a YANG-based protocol?	M	5.29 item d), 46.2, 46.3	Yes [ ]
CNC-S-13	Does the implementation conform to the conditional requirements for use of SRP?	M	5.29 item e), 46.2, 12.32.4	Yes [ ]

Annex B

(normative)

PICS proforma—End station implementations[5]

B.5 Major capabilities

Insert the following row at the end of the table in B.5:

Item	Feature	Status	Reference	Support
CNC-S	Does the implementation support the functionality of a Centralized Network Configuration (CNC) station?	O	5.29, 46.2, A.50	Yes [ ] No [ ]

B.10 Stream Reservation Protocol

Change the following rows of the table in B.10 as shown:

Item	Feature	Status	Reference	Support
SRP-1	Does the implementation support the exchange of Multiple Stream Registration Protocol Data Units (MSRPDUs), using the generic Multiple Registration Protocol Data Unit (MRPDU) format defined in 10.8 to exchange MSRP-specific information, as defined in 35.2.2.8.1, 35.2.2.9.1, and 35.2.2.10.1?	M	5.4.4, 10.8, 35.2.2.8.1, 35.2.2.9.1, 35.2.2.10.1	Yes [ ]
SRP-5	Does the ProtocolVersion used for the implementation of MSRP take the hexadecimal value ~~0x00~~ 0x01?	M	35.2.2.3	Yes [ ]
SRP-9	Does the implementation encode the values in FirstValue fields in accordance with the definition in 35.2.2.8, 35.2.2.9, and 35.2.2.10?	M	35.2.2.8, 35.2.2.9, 35.2.2.10	Yes [ ]
SRP-10	Does the Talker implementation populate the Accumulated Latency with a reasonable, nonzero value?	M	35.2.2.8.6, 35.2.2.10.10	Yes [ ]
SRP-11	Does the implementation update the Failure Information end station MAC address and Code when a Talker Failed is declared?	M	35.2.2.8.7, 35.2.2.10.9, 35.2.2.10.12	Yes [ ]
SRP-13	Is talkerPruning and MMRP supported?	O	~~35.2.4.3.1~~ 35.2.4.3.2	Yes [ ] No [ ]

[5]*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.

Insert the following rows into the table in B.10 in numeric order:

Item	Feature	Status	Reference	Support
SRP-22	Does the implementation support Stream transformation in end station?	O	35.2.2.10.5	Yes [ ] No [ ]
SRP-23	Does the implementation declare the Domain attribute prior to Talker/ Listener attributes?	M	35.2.2.9	Yes [ ]

Insert the following text (Annex U) after Annex T [see subsequent instructions for the "Bibliography" annex (now Annex V)]:

Annex U

(informative)

TSN configuration examples

This annex provides implementation examples for Time-Sensitive Networking (TSN) configuration (Clause 46).

The examples in this annex are not intended to be comprehensive, but to serve as informative background to assist in understanding of Clause 46.

U.1 Examples for time-aware talker

One of the goals of TSN configuration is avoid forcing the user's application to be subservient to the network's configuration. The user describes what it does (e.g., TrafficSpecification) and what it needs from the network (e.g., UserToNetworkRequirements). The user also describes the configuration that it is willing to accept from the network (e.g., InterfaceCapabilities), but the goals are to keep that configuration to a minimum and to avoid topics that change aspects of the user's application, such as the timing of the application's execution and interaction with the physical world (e.g., sensors, actuators).

Aligned with these goals, it is possible for the Talker to transmit frames in a manner that is not aware of time synchronization protocols like IEEE Std 802.1AS. Using the time-aware features of TrafficSpecification and InterfaceConfiguration, it is also possible for a Talker to use awareness of time synchronization for transmission of frames. Nevertheless, TSN configuration avoids assumptions of a specific implementation of time-aware transmit, and TSN configuration avoids changes to application timing.

In order to demonstrate these concepts, consider an example in which Talkers and Listeners use IEEE Std 802.1AS to execute synchronized application loops at an interval of 500 µs. The code for the application consists of two stages: a 250 µs computation stage that generates data and a 250 µs stage to transmit that data in TSN frames. The MaxLatency requirements can span one or more iterations of the application loop.

This example uses IEEE Std 802.1AS for time synchronization. The application begins execution on September 13, 2020, at 12:26:40 pm TAI, which corresponds to 1 600 000 000 000 000 µs in IEEE 802.1AS time.

The application loops begin in alignment with IEEE 802.1AS time; in other words, the 250 µs transmit stage begins at a base offset of 250 µs in IEEE 802.1AS time. For example, after the application has been running for 6 s, a sequence of transmit stages can execute at 1 600 000 006 000 750 µs, 1 600 000 006 001 250 µs, 1 600 000 006 001 750 µs, and so on.

This example focuses on a single Talker that transmits two Streams, J and K. The application generates a single frame of data for J and/or K at arbitrary times in the computation. In other words, during a single iteration data for K can follow J, and in the next iteration data for J can follow K, and in the next iteration data can be J only.

The frame for Stream J uses 120 µs in time. The frame for Stream K uses 80 µs in time. This implies a 100 Mb/s LAN technology.

U.1.1 Using enhancements for scheduled traffic

One option for implementation of this time-aware Talker is the enhancements for scheduled traffic (8.6.8.4). The Talker has two queues for transmit: one for TSN frames and another for non-TSN frames (e.g., best effort). The scheduled traffic enhancements use IEEE 802.1AS time to open and close gates for each queue (traffic class).

The enhancements of 8.6.8.4 schedule traffic per queue, and not per stream. Given the preceding application assumptions, therefore, the frame for Stream J can precede the frame for Stream K, and in the next cycle K can precede J.

For a time-aware Talker, the Jitter of TrafficSpecification (46.2.3.5) represents the variance in time for the single Stream. Since the enhancements of 8.6.8.4 are per queue, the Jitter must include the variance that is introduced by other Streams from this Talker.

Using the scheduled traffic enhancements of 8.6.8.4, the time-aware Talker needs an open TSN window of 120+80=200 µs for both J and K. The network (CNC) can locate that TSN window from 250 µs to 450 µs through 300 µs to 500 µs. The Talker uses the TimeAwareOffset returned in InterfaceConfiguration (46.2.5.3) to determine where to locate the TSN window.

For this implementation, the TrafficSpecifications are as follows:

- Stream J
  - MaxFrameSize = (120 µs in time)
  - MaxFramesPerInterval = 1
  - Interval = 500 µs
  - Jitter = 40 µs (Stream K can occur before J, such that J is 80 µs in window. Stream K can occur after J, such that J is 0 µs in window. Jitter represents the midpoint of this variance. Use the returned TimeAwareOffset-Jitter as the start of the TSN window.)
  - EarliestTransmitOffset = 250 µs + Jitter = 290 µs
  - LatestTransmitOffset = 500 µs – <MaxFrameSize in time> – Jitter = 340 µs
- Stream K
  - MaxFrameSize = (80 µs in time)
  - MaxFramesPerInterval = 1
  - Interval = 500 µs
  - Jitter = 60 µs (Stream J can occur before or after K)
    EarliestTransmitOffset = 250 µs + Jitter = 310 µs
  - LatestTransmitOffset = 500 µs – <MaxFrameSize in time> – Jitter = 360 µs

Assume that the CNC decides to use EarliestTransmitOffset, presumably because that can be aligned with the schedules in Bridges. This results in InterfaceConfigurations of

- Stream J
  - TimeAwareOffset = 290 µs
- Stream K
  - TimeAwareOffset = 310 µs

The Talker subtracts the Jitter from these TimeAwareOffset values and uses the lower of the results to configure AdminBaseTime and OperBaseTime of 8.6.8.4. The resulting configuration of the scheduled traffic enhancements of 8.6.8.4 is shown in Figure U-1 along with example traffic.

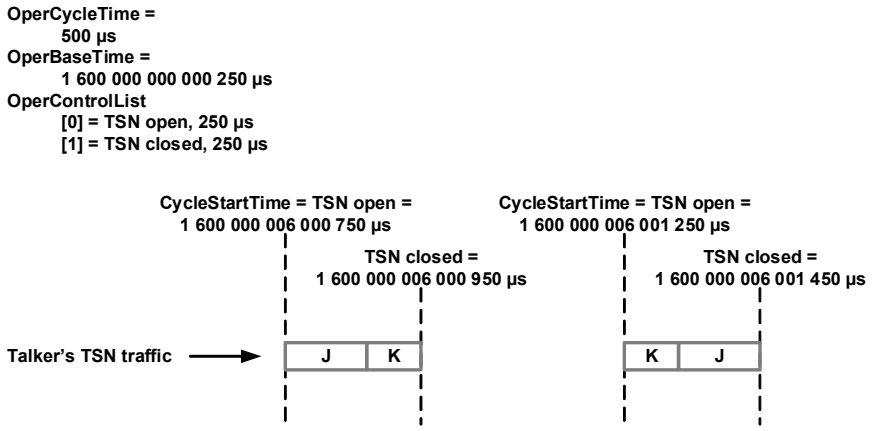

Figure U-1—Example of enhancements for scheduled traffic

NOTE—For this example, the Talker could also decide to use OperBaseTime of 0 µs and swap OperControlList elements 0 and 1 (i.e., 250 µs TSN closed followed by 250 µs TSN open). That configuration is equivalent from the network perspective.

U.1.2 Using strict priority

If the Talker has no hardware to assist in scheduling of traffic, it can take advantage of its application timing for TSN configuration of its Streams. This implementation assumes that the Talker has two queues for transmit, but each queue uses strict priority alone for transmission selection (8.6.8.1). The TSN queue uses a higher priority traffic class than the non-TSN queue.

For this implementation, assume that the Talker disables the TSN queue during its computation stage, but continues to place frames for J and K into the TSN queue. When the Talker transitions to its transmit stage (i.e., 250 µs base offset in IEEE 802.1AS time), it enables the TSN queue for transmit.

When the TSN queue is enabled, a non-TSN frame (e.g., best effort) can interfere such that it has just begun its transmit. For 100 Mb/s full-duplex IEEE Std 802.3 and a maximum frame size of 1522, this interference is 123.36 µs. This interference must be incorporated into the Jitter of TrafficSpecification, much like the Jitter of Streams J and K in U.1.1.

This implementation has no flexibility in the location of its TSN window, which always starts at a 250 µs offset.

For this implementation, the TrafficSpecifications are as follows:

- Stream J
  - MaxFrameSize = (120 µs in time)
  - MaxFramesPerInterval = 1
  - Interval = 500 µs
  - Jitter = (80 µs + 123.36 µs) / 2 = 101.68 µs
    (Stream K and max non-TSN can occur before J.)
  - EarliestTransmitOffset = 250 µs + Jitter = 351.68 µs

    - LatestTransmitOffset = 250 µs + Jitter = 351.68 µs
      (No flexibility. TSN window always starts at 250 µs.)
- Stream K
    - MaxFrameSize = (80 µs in time)
    - MaxFramesPerInterval = 1
    - Interval = 500 µs
    - Jitter = (120 µs + 123.36 µs) / 2 = 121.68 µs)
      EarliestTransmitOffset = 250 µs + Jitter = 371.68 µs
    - LatestTransmitOffset = 250 µs + Jitter = 371.68 µs

This results in the following InterfaceConfigurations (or a failure):

- Stream J
    - TimeAwareOffset = 351.68 µs
- Stream K
    - TimeAwareOffset = 371.68 µs

Due to the high Jitter and inflexibility of the scheduling window, this implementation has the potential for more Stream configuration failures as compared to the implementation described in U.1.1.

U.1.3 Using per-stream scheduling

Some Talker implementations provide additional hardware assistance for time-aware transmission, such that each individual Stream has its own scheduled gating (i.e., per stream).

One such implementation is similar to the scheduled traffic enhancements of 8.6.8.4, but each Stream has a dedicated transmit queue and gate control list. An alternative implementation has a single transmit queue for TSN traffic, but each frame has an associated timestamp to specify its time for transmit. A variety of implementations are possible. For this example, assume that scheduled transmit is per stream, but do not assume a specific hardware implementation.

Assume that the per-stream scheduling hardware uses IEEE 802.1AS time directly. Therefore, the time-aware transmit has no additional variance beyond what IEEE Std 802.1AS provides, and the Jitter of TrafficSpecification is zero.

Given the application requirements, the CNC can locate Stream J anywhere in the window from 250 µs to 500 µs. The CNC can also locate Stream K anywhere in the window from 250 µs to 500 µs, but it cannot overlap with Stream J.

For this implementation, the TrafficSpecifications are as follows:

- Stream J
    - MaxFrameSize = (120 µs in time)
    - MaxFramesPerInterval = 1
    - Interval = 500 µs
    - Jitter = 0 µs
    - EarliestTransmitOffset = 250 µs + Jitter = 250 µs
    - LatestTransmitOffset = 500 µs – <MaxFrameSize in time> – Jitter = 380 µs
- Stream K
    - MaxFrameSize = (80 µs in time)
    - MaxFramesPerInterval = 1
    - Interval = 500 µs

— Jitter = 0 µs
  EarliestTransmitOffset = 250 µs + Jitter = 250 µs
— LatestTransmitOffset = 500 µs – <MaxFrameSize in time> – Jitter = 420 µs

Assuming that the CNC can locate Stream J at 250 µs and Stream K immediately after, this results in the following InterfaceConfigurations:

— Stream J
  — TimeAwareOffset = 250 µs
— Stream K
  — TimeAwareOffset = 370 µs

Due to the zero Jitter and improved flexibility of the scheduling window, this implementation has the potential for more Stream configuration success as compared to the implementation described in U.1.1. In addition, given that each Stream has a distinct offset in time, this implementation enables the CNC to configure the scheduled traffic enhancements of 8.6.8.4 in Bridges such that multiple Streams do not overlap upon ingress from multiple Ports. In turn, the overall variance in latency is reduced.

U.2 Example of workflow for fully centralized models

This subclause provides an example workflow for the centralized TSN configuration models of Clause 46. The example is not intended to be comprehensive, but to help the reader understand how aspects of the configuration can be addressed (e.g., TSN domain detection and enforcement).

The example assumes the fully centralized model (46.1.3.3) with a single Centralized User Configuration (CUC) entity and a single Centralized Network Configuration (CNC) entity.

Many user-level protocol standards exist for the purpose of discovering and configuring end stations in order to design a distributed application. The application is usually specific to a particular market segment (e.g., industrial automation, professional audio/video). A centralized software entity is used for the application's design, possibly including programming of software in end stations (e.g., industrial Programmable Logic Controller). The centralized entity uses the user-level protocol to communicate with the relevant end stations for its application. When TSN configuration is applied to these existing standards, the goal is to integrate TSN functionality without significant changes. The existing application design software is represented by the CUC concept, and the end stations are represented by the Talker and Listener concepts. This example does not designate a specific user-level protocol, but it does assume the existence of a single CUC.

The CNC discovers capabilities of network infrastructure (e.g., Bridges) and configures those features.

The example provides a workflow as a framework for describing each step in the TSN configuration procedure. The steps do not necessarily need to occur sequentially.

NOTE—Although the example assumes a single CUC, its CNC workflow (i.e., step 4 through step 8 below) can be applied to other examples that assume multiple CUCs, such as the following:

a) Multiple application design tools are used (i.e., multiple user-level protocols), and therefore multiple CUCs are used with a single CNC.
b) No application design tool is used, and instead each CUC represents a single Talker or Listener that communicates directly with the CNC.
c) Each CUC represents a single Talker or Listener that uses the Nearest Bridge as a proxy to the CNC (i.e., centralized network/distributed user model).

The workflow for this example consists of the following steps:

1) **CUC discovers Talker and Listener end stations.**

   The protocol used by the CUC is outside the scope of this standard. When the communication between CUC and end stations uses IP, protocols such as mDNS (IETF RFC 6762 [B83]) are sometimes used.

2) **CUC reads the capabilities of each end station.**

   This step includes TSN capabilities, but also many other capabilities that are not related to network technology. For example, if the end station is a sensor, the CUC reads the capabilities of the sensor as it applies to the physical world. The CUC also reads the frame size that is needed to transfer the sensor's measured value and associated diagnostics.

   Although the communication between CUC and end station is not directly related to Clause 46, the CUC needs to obtain the following information:
   - EndStationInterfaces: Number of Ports and MAC address of each Port
   - InterfaceCapabilities: IEEE 802.1 capabilities of each Port
   - TrafficSpecification.MaxFrameSize and MaxFramesPerInterval: Size of end station's data

   The user-level protocol used by the CUC is outside the scope of this standard.

3) **End-user designs the distributed application using the CUC.**

   The end user of the CUC decides which end stations communicate with one another as part of the distributed application. In other words, in terms of this standard, the CUC is used to design the Streams, including selection of the Talker and Listeners for each Stream. The CUC is also used to design the application's timing requirements.

   The CUC creates the following information for each Stream, and its end stations are not directly aware of this information:
   - StreamID: This identifier is primarily used between CUC and CNC.
   - StreamRank: The importance of each Stream is related to its use in the application.
   - UserToNetworkRequirements: MaxLatency is tied to application timing requirements.

   The periodic execution of application code is designed using the CUC. This translates to the interval (i.e., period, rate) at which a Talker transmits its data. The CUC sends this interval to its end stations using the user-level protocol.

4) **CNC discovers the physical network topology.**

   To discover the physical links between end stations and Bridges, the CNC uses IEEE Std 802.1AB (LLDP) along with a remote management protocol.

5) **CNC reads the TSN capabilities of each Bridge.**

   The CNC uses a remote management protocol to read the TSN capabilities of each Bridge.

   Whereas the previous step used the MIB/YANG of IEEE Std 802.1AB, this step uses the MIB/YANG of standards for TSN, such as IEEE Std 802.1Q, IEEE Std 802.1AS, and IEEE Std 802.1CB. The CNC uses the Chassis (Bridge) and Port identifications from the IEEE 802.1AB MIB/YANG to find the corresponding Port capabilities in the MIB/YANG of other IEEE 802.1 standards.

For example, the CNC will read the Bridge Delay (12.32.1) and Propagation Delay (12.32.2) from each Bridge in order to compute AccumulatedLatency (for step 9).

The CNC of this fully centralized model coexists with the fully distributed model (46.1.3.1). If the CNC finds a Bridge with msrpEnabledStatus TRUE (12.22.1) and MRP externalControl FALSE (12.32.4.1), the CNC avoids use of MSRP's VLAN ID (12.22.2, 12.22.4) for Streams configured by the CNC. The CNC also avoids use of MSRP's Priority (12.20.4) and associated traffic class. Although there might be methodologies for sharing the same resources (VLANs and traffic classes) between both models, this CNC takes the simpler approach of keeping the resources separate.

6) **CUC sends Talker/Listener groups to the CNC.**

The CUC requests configuration of Streams by sending Join requests (46.2.2) that specify the Talker group and Listener groups for each Stream.

The EndStationInterfaces group of each Talker and Listener identifies the associated end station in the physical topology, using the end station's MAC address. The MAC address is persistent (i.e., does not change due to software or protocols) and is assumed to be unique within the network, which serves as excellent identification in the physical topology. For this example, assume that end stations use MAC address as their LLDP Chassis ID, such that the CNC can associate each Talker/Listener group to its neighboring Bridge Port.

7) **CNC configures TSN domain(s).**

Using the information learned in step 4 through step 6, the CNC knows the Bridges in the path(s) between each Talker and its Listeners. The CNC also knows the Bridge Ports that do not transmit frames for a Stream. This effectively defines the TSN domain for the collection of Streams.

The CNC uses destination MAC address and VLAN ID as tools to configure TSN features within the TSN domain. The InterfaceConfiguration group provides a mechanism for the CNC to allocate destination MAC addresses and VLAN IDs from its own pool and to provide the allocated values to the CUC (and the end stations). If end stations do not support InterfaceConfiguration, the CNC learns the destination MAC addresses and VLAN IDs from the DataFrameSpecification of each Stream.

i) Inside the TSN domain

When Bridges in the TSN domain are using spanning tree protocols (e.g., MSTP, ISIS-SPB) to forward Streams, the CNC uses management of the Static Filtering Entries (12.7) in order to avoid spanning tree protocols updating the active topology of the VLAN used by Streams.

When the CNC uses IEEE Std 802.1CB for redundant paths for Streams, the CNC uses management of Static Trees (12.32.3) and the associated TE-MSTID in order to avoid spanning tree operation on the VLAN ID used by Streams. The CNC uses management of IEEE Std 802.1CB to configure the redundant paths in Bridges between each Talker and its Listeners.

ii) Outside the TSN domain

The CNC uses management of Static Filtering Entries (12.7) to prevent transmit of TSN frames to Bridge Ports outside the TSN domain.

The CNC uses management of the Priority Regeneration Table (12.6) to prevent interference by non-TSN frames received on Bridge Ports outside the TSN domain, in that the priority of the

non-TSN frame can be downgraded to zero. This is similar to the use of the Priority Regeneration Override Table (12.20.3) by SRP (6.9.4).

The accuracy/precision of IEEE Std 802.1AS degrades slightly for each Bridge that transfers time, so it makes sense for the CNC to limit the size of the IEEE 802.1AS domain as much as possible. As its default mode of operation, the CNC of this example uses management of IEEE Std 802.1AS to disable operation on Ports outside of the TSN domain. For example, assume that the CNC detects two distinct TSN domains G and H with respect to Talkers and Listeners (i.e., no Stream from G to H or vice versa). There is a Bridge between TSN domains G and H, and that Bridge supports IEEE Std 802.1AS. Rather than use a single IEEE 802.1AS domain that spans G and H, the CNC disables IEEE Std 802.1AS on the Ports between G and H; this action effectively creates two distinct IEEE 802.1AS domains, one for each TSN domain.

8) **CNC configures TSN features for Streams.**

The CNC uses the management of IEEE Std 802.1Q to configure TSN features in the path(s) from Talker to Listeners. Examples include enhancements for scheduled traffic (8.6.8.4), frame preemption (6.7.2), and the credit-based shaper (8.6.8.2).

9) **CNC returns the Status of each Stream to the CUC.**

This step includes the success/failure of each Stream's configuration, the AccumulatedLatency, and the InterfaceConfiguration for each end station.

10) **CUC configures each end station.**

If one or more Streams fail configuration, the CUC might decide to adjust its requirements and try again (i.e., return to step 3).

Assuming that the Stream status is sufficient to proceed, the CUC configures each end station to prepare it for application execution. Although the communication between CUC and end station is not directly related to Clause 46, the CUC provides the following information for each Stream:

— InterfaceConfiguration: Provides the destination MAC address and/or VLAN ID (if needed).

11) **CUC executes the distributed application.**

If there are other CUCs for other applications, the CNC can continue to configure Bridge resources for those while this application runs.

12) **CUC stops the distributed application.**

The CUC can send Leave requests (46.2.2) for Streams that are no longer needed.

If the application is changed, this workflow can return to step 1, step 2, or step 3.

Change the designation for the Bibliography annex to Annex V:

Annex V

(informative)

Bibliography

Insert the following bibliography references into Annex V in alphanumeric order:

[B79] IEC 62439-3:2016, Industrial communications networks — High availability automation networks — Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR).

[B80] IETF RFC 791, Internet Protocol—DARPA Internet Program Protocol Specification, Sept. 1981.[6]

[B81] IETF RFC 2460, Internet Protocol, Version 6 (IPv6) Specification, Dec. 1998.

[B82] IETF RFC 6241, Network Configuration Protocol (NETCONF), June 2011.

[B83] IETF RFC 6762, Multicast DNS, Feb. 2013.

[B84] IETF RFC 7223, A YANG Data Model for Interface Management, May 2014.

[B85] IETF RFC 8040, RESTCONF Protocol, Jan. 2017.

[6] IETF documents (i.e., RFCs) are available from the Internet Engineering Task Force (https://www.rfc-editor.org/).

Consensus

WE BUILD IT.

Connect with us on:

Facebook: https://www.facebook.com/ieeesa

Twitter: @ieeesa

LinkedIn: http://www.linkedin.com/groups/IEEESA-Official-IEEE-Standards-Association-1791118

IEEE-SA Standards Insight blog: http://standardsinsight.com

YouTube: IEEE-SA Channel

IEEE
standards.ieee.org
Phone: +1 732 981 0060 Fax: +1 732 562 1571