IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks

IEEE Computer Society

Sponsored by the
Microprocessor Standards Committee

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

IEEE Std 1722™-2016
(Incorporates
IEEE Std 1722-2016 Cor1-2016
Revision of
IEEE Std 1722-2011)

IEEE Std 1722™-2016
(Revision of IEEE Std 1722-2011)

IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks

Sponsor

Microprocessor Standards Committee
of the
IEEE Computer Society

Approved 7 December 2016

IEEE-SA Standards Board

Abstract: The protocol, device discovery, connection management, and device-control procedures used to facilitate interoperability between audio- and video-based end stations that use IEEE 1722–based streams on IEEE 802–based networks are specified.

Keywords: advanced encryption standard (AES), bridged local area network (LAN), clock recovery, elliptic curve cryptography (ECC), encryption, IEC 61883, IEEE 802.1™ AVB protocols, IEEE 802.1AS™, IEEE 802.1BA™-2011, IEEE 802.1Q™-2014, IEEE 1722™-2011, National Institute of Standards and Technology (NIST), QoS, serial digital interface (SDI), SMPTE ST 2022-6, time-sensitive media streaming, Time-Sensitive Networking (TSN), time synchronization

---

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

Copyright © 2016 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 16 December 2016. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-1-5044-0802-8 STD20865
Print: ISBN 978-1-5044-0803-5 STDPD20865

IEEE prohibits discrimination, harassment, and bullying.
For more information, visit http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html.
No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

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 Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

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. Volunteers 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 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

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

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

Every IEEE standard is subjected to review at least every 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 http://ieeexplore.ieee.org/xpl/standards.jsp 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 http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://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 http://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 IEEE standard was completed, the Audio/Video Bridging Layer2 Transport Working Group had the following membership:

David Olsen, *Chair*
Matthew Xavier Mora, *Vice Chair*
Don Pannell, *Secretary*
Aaron Gelter, *Editor*
Jeffrey Daniel Koftinoff, *Editor*

Clause Editors
Gordon Bechtel, *Clause 9, Clause 10*
Rob Silfvast, *Clause 11, Clause 12*
Ethan Grossman, *Clause 8, Annex K*
Willem-Jan Dirks, *Clause 12, Clause 13, Annex J*
Ashley Butterworth, *Clause 15, Annex I*

Frank Bähren
Gordon Bechtel
Alex Belits
Robert Boatright
Kyle Carbone
Tim Carmean
Rodney Cummings
Rob Currie
Jim Czekaj
Edison David
Rebeca Delgado
Jan Eveleens
Nandhini Ganesan
Jan-Willem de Groot
Bob Edge
Thomas Edwards
Don Franks
Amanda Ge
Craig Gunther
Robbie Gurdan
Oliver Hartkopp
Chris Healy
Joern Henneberg
Clint Herran
Mark Hu
Jeffrey Hutchings
Girault Jones Jr.
Haysam Kadry
Andreas Kern
Jochen Klaus-Wagenbrenner
Andre Labelle
Paul Lambert
Jim Lawlis
Herbert Leuwer
Steve Matovski
Greg Nelson
Josef Nöbauer
Chris Pane
Greg Schlechter
Harkirat Singh
Jose Solis-Marquez
Simon Song
Kevin Stanton
Thilo Streichert
Ryan Tabrah
Michael Johas Teener
Max Turner
Stephen Turner
Kieran Tyrrell
Yu-Ting Wu
Mike Ye
Helge Zinner

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

Stefan Aust
Gordon Bechtel
Christian Boiger
Ashley Butterworth
Juan Carreon
Yesenia Cevallos
Minho Cheong
Keith Chow
Rodney Cummings
Richard Edgar
Yukihiro Fujimoto
Aaron Gelter
James Gilb
David Gregson
Ethan Grossman
Randall Groves
Craig Gunther
Marco Hernandez
Werner Hoelzl
Noriyuki Ikeuchi
Atsushi Ito
Raj Jain
Michael Johas Teener
Adri Jovin
Piotr Karocki
Stuart Kerry
Yongbum Kim
Jeff Koftinoff
Arthur H. Light
Michael Lynch
Syam Madanapalli
Roger Marks
Arthur Marris
Matthew Mora
David Olsen
Chris Pane
Arumugam Paventhan
Venkatesha Prasad
Alon Regev
Bartien Sayogo
Stephen Schwarm
Thomas Starai
Eugene Stoudenmire
Walter Struppler
David Thompson
Kieran Tyrrell
Dmitri Varsanofiev
Khurram Waheed
Hung-Yu Wei
Oren Yuen
Daidi Zhong

Copyright © 2016 IEEE. All rights reserved.

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

Thomas Alexander
Stefan Aust
Gordon Bechtel
Ashley Butterworth
Juan Carreon
David Chalupsky
Keith Chow
Richard Doyle
Marc Emmelmann
Yukihiro Fujimoto
Aaron Gelter
James Gilb
Joel Goergen
Ethan Grossman
Randall Groves
Craig Gunther
Marco Hernandez
Werner Hoelzl
Noriyuki Ikeuchi
Atsushi Ito
Michael Johas Teener
Piotr Karocki
Ralph Kearfott
Stuart Kerry
Yongbum Kim
Jeff Koftinoff
James Lepp
Jon Lewis
Arthur Marris
Michael Montemurro
Matthew Mora
Michael Newman
Paul Nikolich
David Olsen
Stephen Palm
Venkatesha Prasad
Alon Regev
Maximilian Riegel
Frank Schewe
Veselin Skendzic
Thomas Starai
Walter Struppler
David Thompson
Kieran Tyrrell
Stephen Webb
Colin Whitby-Strevens
Oren Yuen
George Zimmerman

When the IEEE-SA Standards Board approved this standard on 7 December 2016, it had the following membership:

John D. Kulick, *Chair*
Jon Walter Rosdahl, *Vice Chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Jean-Philippe Faure
J. Travis Griffith
Gary Hoffman
Michael Janezic
Joseph L. Koepfinger*
David J. Law
Hung Ling
Andrew Myles
T. W. Olsen
Glenn Parsons
Ronald C. Petersen
Annette D. Reilly
Stephen J. Shellhammer
Adrian P. Stephens
Yatin Trivedi
Philip Winston
Don Wright
Yu Yuan
Daidi Zhong

*Member Emeritus

Copyright © 2016 IEEE. All rights reserved.

Introduction

This introduction is not part of IEEE Std 1722-2016, IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks.

Increasingly, entertainment media are digitally transported. Streaming audio/video and interactive applications over local area networks is becoming more common.

This standard builds on the work done by the IEEE 802.1 AVB task group by providing a common audio/video transport protocol capable of supporting the needs of both consumer and professional audio/video applications.

Copyright © 2016 IEEE. All rights reserved.

Contents

1. Overview 1
1.1 Scope 1
1.2 Purpose 2
1.3 Notes about this revision 2

2. Normative references 3

3. Terms, definitions, and notation 5
3.1 Definitions 5
3.2 Reserved fields 8
3.3 Numerical values 8
3.4 Notation of fields and values taken from other documents 8
3.5 Acronyms and abbreviations 11

4. AVTP base protocol 13
4.1 Overview 13
4.2 AVTP network requirements 13
4.3 Timing and synchronization 14
4.4 Header formats 20

5. IEC 61883/IIDC Format 29
5.1 Overview 29
5.2 IEC 61883/IIDC stream data encapsulation 29
5.3 IIDC encapsulation 32
5.4 IEC 61883 encapsulation 39

6. MMA Format 48

7. AVTP Audio Format 48
7.1 Overview 48
7.2 AAF common stream data encapsulation 48
7.3 AAF PCM stream data encapsulation 51
7.4 AAF AES3 stream data encapsulation 56
7.5 AAF timestamps 60
7.6 AAF packetization 61
7.7 AAF and SRP 61

8. Compressed Video Format 63
8.1 Overview 63
8.2 AVTP Presentation Time handling 63
8.3 Compressed Video Stream data encapsulation 64
8.4 MJPEG Video Format 66
8.5 H.264 Compressed Video Format 69
8.6 JPEG 2000 Compressed Video Format 73

9. AVTP Control Format 77
9.1 Overview 77
9.2 Non–Time-Synchronous Control Format header 78
9.3 Time-Synchronous Control Format header 79
9.4 ACF messages 81

Copyright © 2016 IEEE. All rights reserved.

10. Clock Reference Format 110
10.1 Overview 110
10.2 Background 110
10.3 Audio example 110
10.4 Clock Reference Format Data encapsulation 112
10.5 CRF transmission rates 117
10.6 Interaction between CRF timestamps and Media Stream timestamps 118
10.7 Requirements of CRF timestamp values 119
10.8 Relationship between CRF timestamps and stream timestamps 120

11. SDI Video Format 121
11.1 Overview 121
11.2 SDI Video Stream data encapsulation 121

12. Raw Video Format 129
12.1 Overview 129
12.2 Raw Video Stream data encapsulation 130

13. AES Encrypted Format 139
13.1 Overview 139
13.2 AEF data encapsulation 140
13.3 AES-SIV 141

14. Vendor Specific Format 142
14.1 Overview 142

15. Experimental Format 144
15.1 Overview 144

16. ECC Signed Control Format 144
16.1 Overview 144
16.2 ECC Signed Control Data encapsulation 145
16.3 ECSSA DSA EMSA1 SHA256 signature variant 147

17. ECC Encrypted Control Format 148
17.1 Overview 148
17.2 ECC Encrypted Control AVTPDU 148
17.3 Elliptic curve variant 1 algorithm 149

Annex A (informative) Bibliography 151

Annex B (normative) MAC Address Acquisition Protocol 153
B.1 Overview 153
B.2 Protocol Message Format 154
B.3 Protocol specification 157
B.4 Reserved MAAP MAC addresses 163

Annex C (informative) IEEE 802.3 Media-specific encapsulation 164
C.1 Introduction 164
C.2 IEEE Std 802.3 encapsulation 164
C.3 Other considerations 168

Annex D (normative) MA-L usage 169

Annex E (informative) Diagnostics and counters 170
E.1 Audio/Video Transport Protocol (AVTP) stream counters 170
E.2 AVTP counter definitions 171

Copyright © 2016 IEEE. All rights reserved.

Annex F (normative) Protocol Implementation Conformance Statement (PICS) proforma....173
F.1 Introduction....173
F.2 Abbreviations and special symbols....173
F.3 Instructions for completing the PICS proforma....174
F.4 PICS proforma for IEEE Std 1722....176
F.5 Major capabilities and options....177
F.6 AVTP Protocol....177
F.7 61883 Protocol....178
F.8 AVTP Audio Format....179
F.9 MMA Format....180
F.10 Compressed Video Format....181
F.11 AVTP Control Format....182
F.12 Clock Reference Format....184
F.13 SDI Video Format....185
F.14 Raw Video Format....186
F.15 AES Encrypted Format....187
F.16 Vendor Specific Format....187
F.17 ECC Signed Control Format....187
F.18 ECC Encrypted Control Format....188
F.19 MAC Address Acquisition Protocol....189
F.20 IP Encapsulation....189

Annex G (informative) Example event field usage....190
G.1 Overview....190
G.2 Synchronized format changes in a automotive environment....190
G.3 Synchronized external event indication....191

Annex H (informative) Content protection....192

Annex I (normative) AVDECC stream formats....193
I.1 Overview....193
I.2 Stream formats....193

Annex J (normative) IP encapsulation....210
J.1 Overview....210
J.2 Encapsulation styles....210
J.3 IP AVTPDU Format....210
J.4 Source port....211
J.5 Destination port....211
J.6 Destination address....212
J.7 Bandwidth reservation....212
J.8 Clock synchronization....212

Annex K (informative) AES3 and AVTP Audio Format....213
K.1 Definitions....213
K.2 AES3 block start....215
K.3 Bit ordering in AES3/AES10 compared with AES47/AAF....215
K.4 Audio sample, preamble, and ancillary data field order in AAF....216
K.5 The Validity (V) bit....216
K.6 The Parity (P) bit....216
K.7 Bit ordering of ancillary data....217
K.8 SMPTE ST 337 data bursts....217
K.9 Using AVDECC to convey AES3 stream data types....217
K.10 Other considerations....219
K.11 Other references....220

Copyright © 2016 IEEE. All rights reserved.

IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks

IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers 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.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/IPR/disclaimers.html.

1. Overview

Increasingly, entertainment media are digitally transported. Streaming audio/video and interactive applications over bridged local area networks (LANs) need to have comparable real-time performance with legacy analog distribution. There is significant end-user and vendor interest in defining a simple yet common method for handling real-time audio/video suitable for consumer electronics, professional audio/video applications, etc. Technologies such as IEEE Std 1394™,[1, 2] Bluetooth®, and USB exist today, but each has their own encapsulation, protocols, timing control, etc., such that building interworking functions is difficult. The use of a common audio/video transport over multiple IEEE 802™ network types will realize operational and equipment cost benefits. By ensuring that all IEEE 802 wired and wireless devices share a common set of transport mechanisms for time-sensitive audio, video, and control streams, the effort of producing interworking units between IEEE 802 devices and other digital networks is lessened.

1.1 Scope

This standard specifies the protocol, data encapsulations, and synchronization procedures used to enable interoperability between time-sensitive audio, video, and control applications using the quality of service capabilities provided by IEEE 802 Time-Sensitive Networking standards.

[1]The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.
[2]IEEE publications are available from The Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

Copyright © 2016 IEEE. All rights reserved.

1.2 Purpose

This standard facilitates interoperability between end stations that transport time-sensitive media across LANs providing time synchronization, latency, and bandwidth services by defining additional packet format protocols, synchronization mechanisms, and diagnostic counters.

1.3 Notes about this revision

1.3.1 Compatibility

Although some field definitions have changed from those defined in the previous version of this standard, this revision of the document is fully backward-compatible with the previous version. For example, the **cd** bit of the Audio Video Transport Protocol Data Units (AVTPDU) common header (see 4.4.3) has been removed, but formats that used this bit in the previous revision have been updated to have the most-significant bit of the **subtype** field set, retaining binary compatibility.

1.3.2 Use of TSN and AVB

With the expansion of use cases to industrial control and other applications that are not audio or video in nature, updated standards have been created that collectively use the moniker Time-Sensitive Networking (TSN). In this document, AVB is considered a subset of the TSN standards, and as such, both names are used. The term AVB is used for audio and video specific applications for which the term Audio Video Bridging (AVB) was originally envisioned, and TSN is used for formats that are not audio or video in nature.

1.3.3 New additions

In addition to the formats defined in the previous version of this standard, this revision adds new formats that add features that were not present in the previous revision or allow for easier implementation of common use cases. A summary of the additions present in this revision is given in Table 1.

Copyright © 2016 IEEE. All rights reserved.

Table 1—New additions in this revision

Addition	Description	Reference
AES-SIV Encrypted Format	Provides a method for encrypting Audio/Video Transport Protocol (AVTP) streams using Advanced Encryption Standard–Synthetic Initialization Vector (AES-SIV).	Clause 11
Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC) stream formats	Annex that describes stream formats that were not available when IEEE Std 1722.1™-2013 was published.	Annex I
AVTP Audio Format	Provides a mechanism for transporting PCM and AES3 audio data over an AVTP network that is optimized for minimum overhead and maximum utilization of stream reservations. AVTP audio also allows for higher channel counts in streams due to lower header overhead compared to the IEC 61883-6 method (see Clause 5).	Clause 7
Clock Reference Format	Allows for the dissemination of timing information to devices using TSN without the use of a media stream.	Clause 10
Compressed Video Format	Makes use of video payload definitions that are defined in Internet Engineering Task Force (IETF) request for comments (RFCs).	Clause 8
Content protection	Annex that gives an overview of using High-bandwidth Digital Content Protection (HDCP) over an AVTP network.	Annex H
Control formats	Allows for both synchronous and non-synchronous delivery of a variety of control messages over TSN.	Clause 9
Diagnostics and counters	Annex that provides suggestions on counters that could be useful for debug.	Annex E
ECC Encrypted Format	Provides a method for encrypting AVTP streams using ECC.	Clause 17
ECC Signed Control Format	Provides a method for signing AVTP streams using ECC.	Clause 16
Event field example usage	Annex that describes possible usages of the application-specific event field bits.	Annex G
Experimental format	A mechanism for testing and developing future enhancements to the IEEE 1722 family of standards.	Clause 15
IP encapsulation	Annex that describes a method for encapsulating any AVTPDU in a User Datagram Protocol (UDP) datagram on an Internet Protocol (IP) network.	Annex J
MA-L usage	Annex that outlines the allocation of addresses within the AVTP-assigned MA-L.	Annex D
PICS proforma	A list of requirements that a device must adhere to in order to be deemed compliant with the standard.	Annex F
Raw Video Format	Provides a direct method for transporting video data (active video only) over an AVTP network.	Clause 12
SDI Video Format	Provides a method for transporting direct mappings of commonly used SDI data streams over an AVTP network.	Clause 11
Vendor Specific Format	Provides a method for vendors to transport proprietary stream formats over AVTP.	Clause 14

2. Normative references

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

AES3-1-2009, AES Standard for Digital Audio—Digital Input-Output Interfacing—Serial Transmission Format for Two-Channel Linearly-Represented Digital Audio Data—Part 1: Audio Content.

AES3-2-2009, AES Standard for Digital Audio—Digital Input-Output Interfacing—Serial Transmission Format for Two-Channel Linearly-Represented Digital Audio Data—Part 2: Metadata and Subcode.

Copyright © 2016 IEEE. All rights reserved.

AES3-3-2009, AES Standard for Digital Audio—Digital Input-Output Interfacing—Serial Transmission Format for Two-Channel Linearly-Represented Digital Audio Data—Part 3: Transport.

AES47-2006 (r2011), AES Standard for Digital Audio—Digital Input-Output Interfacing—Transmission of Digital Audio over Asynchronous Transfer Mode (ATM) Networks.

IEC 61883-1:2008, Consumer Audio/Video Equipment—Digital Interface—Part 1: General.[3]

IEC 61883-4:2004, Consumer Audio/Video Equipment—Digital Interface—Part 4: MPEG2-TS Data Transmission.

IEC 61883-6:2014, Consumer Audio/Video Equipment—Digital Interface—Part 6: Audio and Music Data Transmission Protocol.

IEC 61883-7:2003, Consumer Audio/Video Equipment—Digital Interface—Part 7: Transmission of ITU-R BO.1294 System B.

IEC 61883-8:2013, Consumer Audio/Video Equipment—Digital Interface—Part 8: Transmission of ITU-R Bt.601 Style Digital Video Data.

IEEE Std 754™-2008, IEEE Standard for Floating-Point Arithmetic.[4, 5]

IEEE Std 802.1AS™-2011, IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.

IEEE Std 802.1Q™-2014, IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks.

IEEE Std 1363a™-2004, IEEE Standard Specifications for Public-Key Cryptography—Amendment 1: Additional Techniques.

IEEE Std 1394™, IEEE Standard for High-Performance Serial Bus.

IEEE Std 1722.1™-2013, IEEE Standard for Device Discovery, Connection Management, and Control Protocol for IEEE 1722™–Based Devices.

IIDC2 Digital Camera Control Specification Ver.1.0.0, Jan. 2012.[6]

NIST FIPS-197, Specification for the Advanced Encryption Standard (AES).[7]

RFC 2435, RTP Payload Format for JPEG-Compressed Video.[8]

RFC 5297, Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES).

RFC 5371, RTP Payload Format for JPEG 2000 Video Streams.

---

[3]IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.
[4]IEEE publications are available from The Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[5]The IEEE standards or products referred to in this clause are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.
[6]Available at http://jiia.org/wp-content/themes/jiia/pdf/standard_dl/ngcp/CP-001-2011_1394TA_TS2011001.pdf.
[7]Available from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[8]RFC documents are a product of the IETF (Internet Engineering Task Force) and are available from http://www.rfc-editor.org.

Copyright © 2016 IEEE. All rights reserved.

RFC 6184, RTP Payload Format for H.264 Video.

SMPTE ST 259:2008, Television—SDTV Digital Signal/Data—Serial Digital Interface.[9]

SMPTE ST 274:2008, Television—1920 × 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates.

SMPTE ST 292-1:2012, 1.5 Gb/s Signal/Data Serial Interface.

SMPTE ST 296:2012, 1280 × 720 Progressive Image 4:2:2 and 4:4:4 Sample Structure—Analog and Digital Representation and Analog Interface.

SMPTE ST 337:2008, Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface.

SMPTE ST 338:2015, Format for Non-PCM Audio and Data in AES3—Data Types.

SMPTE ST 425-1:2011, Source Image Format and Ancillary Data Mapping for the 3 Gb/s Serial Interface.

SMPTE ST 2022-6:2012, Transport of High Bit Rate Media Signals over IP Networks (HBRMT).

3. Terms, definitions, and notation

3.1 Definitions

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

4:1:1, 4:2:0, 4:2:2, 4:4:4: Denotes the proportion of luma (Y) to chroma (U and V) components. For example in 4:1:1 there is one U and one V for every four Y components.

4:2:2:4, 4:4:4:4: Denotes the proportion of luma (Y) to chroma (U and V) components similar to 4:2:2 and 4:4:4, but with an extra channel for keying.

audio sample frame: An audio sample frame is a set of audio samples in a stream that are intended to be processed at the same instant. For example in a stereo audio stream an audio sample frame consists of two audio samples.

audio/video bridging (AVB): A set of specifications that allow time-synchronized, low-latency streaming services through IEEE 802 networks, as defined by IEEE Std 802.1BA™. Specifically, this includes the generalized Precision Time Protocol (gPTP) as defined in IEEE Std 802.1AS™, as well as the Stream Reservation Protocol (SRP) and the Forwarding and Queuing Enhancements for Time-Sensitive Streams (FQTSS) as defined in IEEE Std 802.1Q™. For the purposes of Audio/Video Transport Protocol (AVTP), AVB and Time-Sensitive Networking (TSN) are equivalent.

audio/vdeo bridging (AVB) bridge: A bridge that implements AVB standards. *See* **audio/video bridging (AVB)**.

[9]SMPTE publications are available at http://standards.smpte.org.
[10]*IEEE Standards Dictionary Online* subscription is available at:
http://www.ieee.org/portal/innovate/products/standard/standards_dictionary.html.

Copyright © 2016 IEEE. All rights reserved.

Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC): The protocol defined by IEEE Std 1722.1™-2013. As an adjective, it indicates that the modified noun is specified or interpreted in the context of IEEE Std 1722.1-2013.

Audio/Video Transport Protocol (AVTP): The protocol defined by this standard. As an adjective, it indicates that the modified noun is specified in or interpreted in the context of this standard. This protocol includes transportation of data beyond audio and video.

AVDECC Connection Management Protocol: The Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC) Connection Management portion of IEEE Std 1722.1-2013 handles the process of making and breaking connections between stream sinks and stream sources.

AVDECC Discovery Protocol: The Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC) Discovery portion of IEEE Std 1722.1-2013 handles advertising and discovery on a network.

AVDECC Enumeration and Control Protocol: The Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC) Enumeration and Control portion of IEEE Std 1722.1-2013 handles enumeration and control of capabilities, formats, and controls.

AVTP gateway: A device that transports Audio/Video Transport Protocol (AVTP) streams between Time-Sensitive Network and another type of media, for example, an IEEE 1394™ bus.

AVTP stream: An Audio/Video Transport Protocol (AVTP) stream is between one Talker and one or more Listeners.

Bayer grbg, rggb, bggr, gbrg: An arrangement of a color filter array where there is 50% green, 25% red, and 25% blue arranged as grbg, rggb, bggr, or grbg.

Controller Area Network (CAN): A serial communications protocol commonly used in automotive environments to connect control modules, sensors, and other devices. See ISO 11898 [B17][11] for more information.

Controller Area Network with Flexible Data-Rate (CAN FD): An extension of the CAN specification that allows for data rates higher than 1 MBit/s and payloads longer than 8 octets per frame. See ISO 11898 [B17] for more information.

device: Within the context of this standard, a device is a hardware entity capable of participating in Audio/Video Transport Protocol (AVTP), functioning as a Talker, Listener, and/or Controller.

doublet: Two octets of data.

End of Active Video (EAV): A byte sequence defined by ITU-R BT.656 which indicates the end of the active video on a given video line, while also indicating the current field and whether the current line is part of vertical blanking.

FlexRay™: An automotive network communications protocol developed by the FlexRay Consortium. See ISO 10681-1 [B15], [B16], for more information.

frame: A frame is a complete unit of data transmission as transmitted on the network, including media access control (MAC) layer addressing and cyclic redundancy check (CRC).

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

Copyright © 2016 IEEE. All rights reserved.

free-wheel: A clock that is in free-wheel mode continues at its current frequency and phase with no corrections. A clock can be in this mode due to a lack of incoming correction information or because correction information is being disregarded.

Grayscale: A color space that carries only intensity information.

Listener: An end station that is the destination, receiver, or consumer of an AVTP stream.

Local Interconnect Network (LIN®): A serial communications protocol used in distributed automotive applications. See LIN Specification Package—Revision 2.2A for more information.

Media Oriented Systems Transport (MOST®): A streaming and communications protocol used in automotive applications. See MOST Specification Rev. 3.0 E2 07/2010 for more information.

Musical Instrument Digital Interface (MIDI) Manufacturers Association (MMA): The publisher and source of MIDI specifications. The MMA publishes the MMA Payload Specification for AVTP.

Observation Interval: The Observation Interval is equivalent to the Class Measurement Interval as defined in Clause 34 of IEEE Std 802.1Q-2014.

octet: Eight bits of data.

packet: Refers to the IEC 61883 definition. Packet is used to refer to specific features of 61883 that use this term, such as source packet header (SPH) or common isochronous packet (CIP).

quadlet: Four octets of data.

Real-time Transport Protocol (RTP): A standardized set of packet formats designed for transporting audio and video across IP networks.

RGB: An additive color space based on a red, green, blue color model.

Serial Data Interface (SDI): A family of digital video interfaces standardized by SMPTE for use in professional video applications.

Start of Active Video (SAV): A byte sequence defined by ITU-R BT.656 that indicates the start of the active video on a given video line, while also indicating the current field and whether the current line is part of vertical blanking.

Stream Reservation (SR) Class: As defined by Clause 35 of IEEE Std 802.1Q-2014.

StreamID: A 64-bit field that uniquely identifies a stream, as defined in Clause 35 of IEEE Std 802.1Q-2014.

Talker: An end station that is the source or producer of an AVTP stream.

Time-Sensitive Networking (TSN): *See:* **audio/video bridging (AVB)**.

Time-Sensitive Networking (TSN) Bridge: *See:* **audio/video bridging (AVB) bridge**.

XYZ: A color space that uses tristimulus values based on the color matching functions of the CIE 1931 [B14] standard observer.

YCbCr: The color space formed by luminance (Y), chrominance blue (Cb), and chrominance red (Cr).

Copyright © 2016 IEEE. All rights reserved.

YCgCo: The color space formed by luminance (Y), chrominance green (Cg), and chrominance orange (Co).

YCM: A subtractive color space based on a yellow, cyan, and magenta color model.

3.2 Reserved fields

Any field within a data structure that is defined in this specification as reserved is reserved for future standardization. All reserved fields specified in this standard shall be set to a value of zero (0) on transmit and shall be ignored on receive.

Reserved fields in this standard are designated as one of **reserved**, **rsv**, or **r**.

3.3 Numerical values

Decimal, hexadecimal, and binary numbers are used within this document. For clarity, decimal numbers are generally used to represent counts, hexadecimal numbers are used to represent addresses, and binary numbers are used to describe bit patterns within binary fields.

Decimal numbers are represented in their usual 0, 1, 2, ... format. Hexadecimal numbers are represented by a string of one or more hexadecimal (0 to 9 and A to F) digits, followed by the subscript 16. Binary numbers are represented by a string of one or more binary (0, 1) digits in left to right order, where the left-most bit is the most-significant bit and the right-most bit is the least-significant bit, followed by the subscript 2. Thus, the decimal number "26" may also be represented as "$1A_{16}$" or "11010_2".

These notational conventions have one exception: media access control (MAC) addresses and IEEE Organizationally Unique Identifier (OUI)/IEEE Extended Unique Identifier (EUI)/MAC Address Blocks (MA-L, MA-M, MA-S) values are represented as strings of 1-octet hexadecimal numbers separated by hyphens and without a subscript as, for example, "01-80-C2-00-00-15" or "AA-55-11."

3.4 Notation of fields and values taken from other documents

Since this document references the protocols described in other documents, the notational conventions for referenced fields and values have been preserved where possible to avoid any confusion.

a) Fields from IEEE Std 802.1Q-2014: Fields are in all uppercase, no underscores (examples: destination MAC address [DA], source MAC address [SA], tagged protocol identifier [TPID], drop eligible indicator [DEI], and VLAN identifier [VID]).

b) Fields from IEEE Std 1394: Fields in lowercase except for acronyms within the field name with optional underscores (examples: tcode, data_length, and source_ID).

c) Fields from IEC 61883: Fields always starting in uppercase, acronyms in uppercase, abbreviations with uppercase first followed by lowercase, no underscores (examples: data block count [DBC], data block size [DBS], and reserved [Rsv]).

Copyright © 2016 IEEE. All rights reserved.

3.4.1 Bit, octet, doublet, and quadlet ordering

All Audio Video Transport Protocol Data Units (AVTPDUs) consist of an integral number of octets that are numbered starting from 1 and increasing in the order that they are put into a Data Link Service Data Unit. The bits in each octet are numbered from 0 to 7, where 7 is the low-order bit.

When consecutive octets are used to represent a binary number, the lower octet number has the most-significant value. When the encoding of (an element of) an AVTPDU is represented using a diagram in this clause, the following representations are used:

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

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

c) Within an octet, bits are shown with bit 0 to the left and bit 7 to the right.

Figure 1 represents a decimal value of 1, a hexadecimal value of 01_{16}, and a binary value of 00000001_2 with the only bit in the octet set to one of the least-significant bit, bit 7.

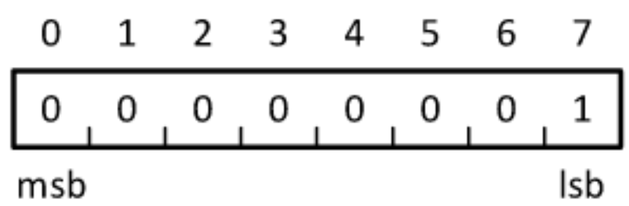


Figure 1—Bit ordering within an octet

A quadlet is a series of four octets. Within a quadlet, the most-significant octet is shown as the left-most octet and the least-significant octet is shown as the right-most, as shown in Figure 2.

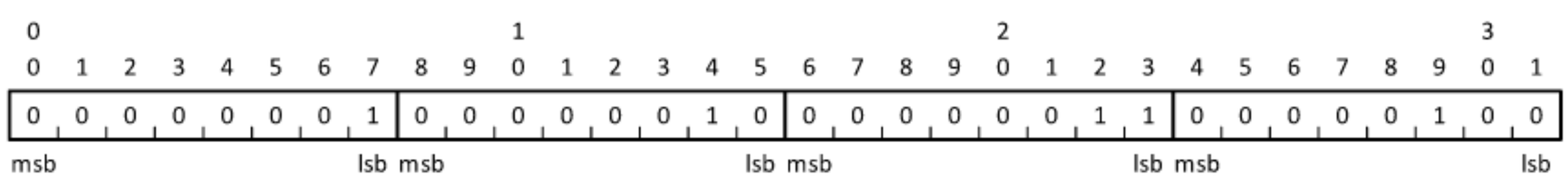


Figure 2—Octet ordering within a quadlet

Figure 2 represents a quadlet holding a four-octet array with values of 1, 2, 3, and 4 decimal for each octet.

A quadlet may contain bit fields of any length between one bit and four octets. When a field spans more than one octet, the point where it spans the octet is shown as a large tick mark as shown in Figure 3.

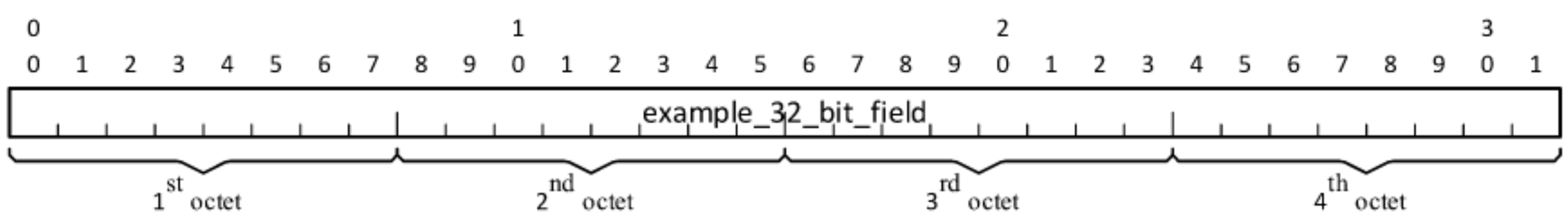


Figure 3—Example 32-bit field diagram

The 64-bit fields that need to be contained in more than one quadlet are shown as a series of eight octets in Figure 4.

Copyright © 2016 IEEE. All rights reserved.

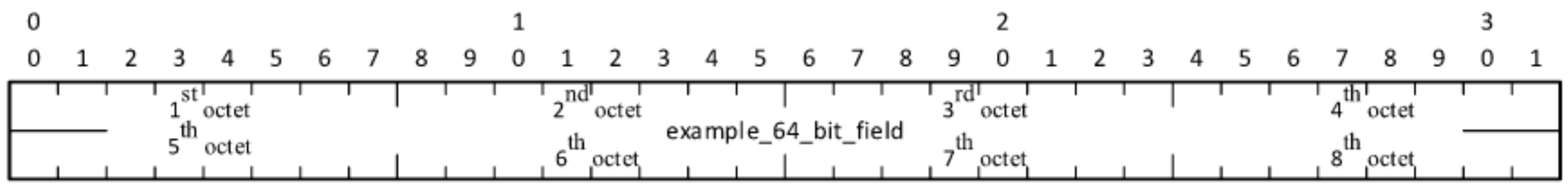


Figure 4—Example 8-octet field diagram

3.4.2 Field value conventions

This document describes values of fields. For clarity, names can be associated with each of these defined values, as illustrated in Table 2. A symbolic name, consisting of uppercase letters with underscore separators, allows other portions of this document to reference the value by its symbolic name, rather than by a numerical value.

Table 2—Wrap field values

Value	Name	Description
0	WRAP_AVOID	Frame is discarded at the wrap point
1	WRAP_ALLOW	Frame passes through wrap points
2 to 3	—	Reserved

A field value of TRUE shall always be interpreted as being equivalent to a numeric value of one (1) unless otherwise indicated. A field value of FALSE shall always be interpreted as being equivalent to a numeric value of zero (0) unless otherwise indicated.

3.4.3 AVTPDU diagram conventions

This document uses black and gray coloring in the AVTPDU diagrams to indicate the focus of discussion within the diagram. Figure 5 shows an example of the usage of black and grayed coloring.

Copyright © 2016 IEEE. All rights reserved.

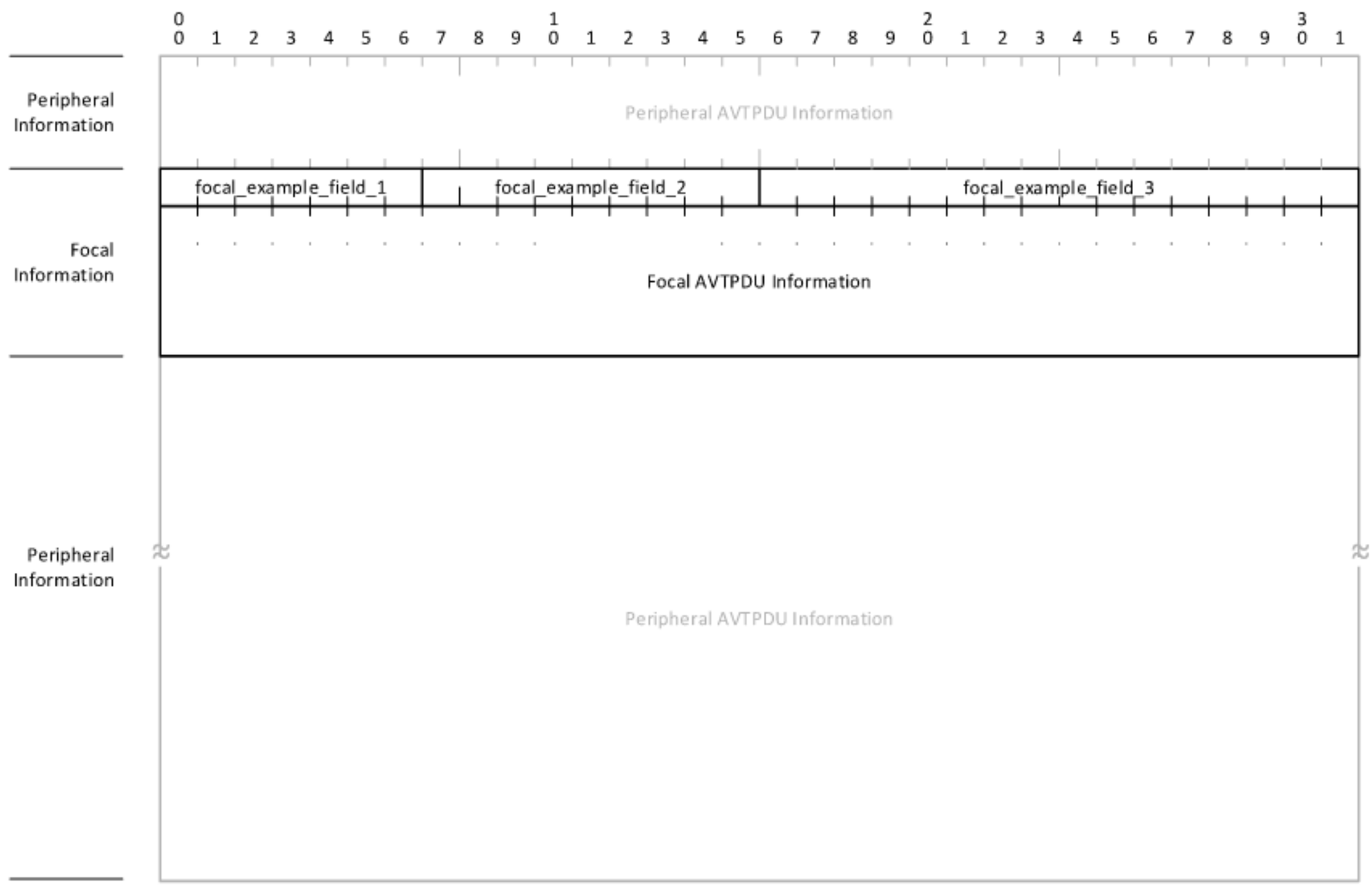


Figure 5—Example diagram showing grayed fields

In Figure 5 items that are focal to the text of the section are shown in black. In this example, the fields **focal_example_field_1**, **focal_example_field_2**, and **focal_example_field_3** would be described and discussed in the section's text. The items labeled "Peripheral AVTPDU Information" are grayed-out to show that they are described elsewhere. The diagram shows them to provide context to the focal items.

3.5 Acronyms and abbreviations

AAF	AVTP Audio Format
ADP	AVDECC Discovery Protocol
AECP	AVDECC Enumeration and Control Protocol
ACMP	AVDECC Connection Management Protocol
AVB	audio/video bridging
AVDECC	Audio/Video Discovery, Enumeration, Connection Management, and Control
AVTP	Audio/Video Transport Protocol
AVTPDU	Audio/Video Transport Protocol Data Unit
CAN	Controller Area Network
CAN FD	CAN with Flexible Data-Rate
CIP	common isochronous packet
DA	destination MAC address
DBC	data block count
DBS	data block size

Copyright © 2016 IEEE. All rights reserved.

DEI	drop eligible indicator
DTCP	digital transmission content protection
DVCR	digital videocassette recorder
EAV	End of Active Video
EUI	IEEE Extended Unique Identifier
FDF	format dependent field
FQTSS	Forwarding and Queuing for Time-Sensitive Streams (Clause 34 of IEEE Std 802.1Q-2014)
FMT	stream format
FN	fraction number
gPTP	generalized Precision Time Protocol (IEEE Std 802.1AS)
HANC	Horizontal Ancillary Data
HDCP	High-bandwidth Digital Content Protection
IIDC	Instrumentation and Industrial Digital Camera
kHz	kilohertz (thousand cycles per second)
LAN	local area network
LIN®	Local Interconnect Network
MAAP	MAC Address Acquisition Protocol
MAC	media access control
MA-S	MAC Address Block Small (formerly OUI-36)
MA-M	MAC Address Block Medium
MA-L	MAC Address Block Large (formerly OUI)
MHz	megahertz (million cycles per second)
MIDI	Musical Instrument Digital Interface
MMA	MIDI Manufacturers Association, Inc.[1]
MMRP	Multiple MAC Registration Protocol
MOST®	Media Oriented Systems Transport
MTU	maximum transmission unit, for this document refers to the maximum length of the MAC Client Data field defined in 3.2.7 of IEEE Std 802.3™-2012
MVRP	Multiple VLAN Registration Protocol (subclause 11.2 of IEEE Std 802.1Q-2014)
NALU	Network Abstraction Layer Unit. The basic transport layer unit for H.264 compressed video. (See 1394 Trade Association, 2004006 [B1].)
OUI	IEEE organizationally unique identifier
PCP	priority code point
PDU	protocol data unit
QPC	quadlet padding count
r	reserved

[1]MIDI Manufacturers Association, Inc., P.O. Box 3173, La Habra, CA 90632-3173, http://www.midi.org.

Copyright © 2016 IEEE. All rights reserved.

rsv	reserved
RTP	Real-time Transport Protocol
SAV	Start of Active Video
SDI	Serial Digital Interface
SID	source identifier
S/PDIF	Sony/Philips Digital Interconnect Format
SRP	Stream Reservation Protocol (Clause 35 of IEEE Std 802.1Q-2014)
SPH	source packet header
sRGB	standard RGB
SYT	synchronization timing
TLV	Type-Length-Value
TPID	tagged protocol identifier
TSN	Time-Sensitive Networking
TSP	Transport Stream Packet
VANC	Vertical Ancillary Data
VDSPC	Video Data Source Packet Count
VID	VLAN identifier
VLAN	Virtual Local Area Network

4. AVTP base protocol

4.1 Overview

Audio Video Transport Protocol (AVTP) specifies a protocol for audio, video, and control data to be transported on a Time-Sensitive Networking (TSN) capable network. AVTP is designed to take advantage of TSN capable networks that support generalized Precision Time Protocol (gPTP), Stream Reservation Protocol (SRP), and Forwarding and Queing Enhancements for Time-Sensitive Streams (FQTSS).

AVTP specifies methods to transport data and timing information so that audio, video, or control content sent by a Talker can be reproduced on Listeners. AVTP streams are sent from a Talker and can be received by one or more Listeners.

AVTP stream data is typically carried directly on an underlying MAC layer such as IEEE Std 802.3 or IEEE Std 802.11™; however, it is also possible to encapsulate AVTP in IP as described in Annex J.

This clause contains the basic assumptions and operation requirements of the protocol.

4.2 AVTP network requirements

All AVTP end stations that send or receive audio, video, or time-sensitive control data are required to support the TSN services provided by gPTP, SRP, and FQTSS. AVTP relies on these services being available to interoperate. AVTP end stations that only support data formats that are not time-sensitive are

Copyright © 2016 IEEE. All rights reserved.

not required to support TSN services. The behavior of a Talker, Listener, or bridge on the network that forwards AVTP audio, video or time-sensitive control data and does not support the TSN protocols is out of the scope of this standard.

AVTP makes use of gPTP to provide a network-wide time base that can be used to convey timing information from a Talker to Listener(s).

AVTP makes use of SRP and FQTSS to provide reliable network delivery with bounded network latency for transporting data from Talker to Listener(s).

AVTP data can be sent from Talker to Listener(s) either directly or forwarded by TSN bridges to the Listener(s).

4.3 Timing and synchronization

4.3.1 General

AVTP defines a presentation time to achieve timing synchronization between a Talker and Listener(s); it represents in nanoseconds the gPTP time when the data contained in the AVTPDU is to be available to the AVTP Listener(s).

AVTP Presentation Time is used as a reference to synchronize any necessary media clocks and to determine when the payload of a stream is to be presented to the time-sensitive application. Media clocks vary with audio/video types therefore the exact usage of the AVTP Presentation Time is media format dependent.

AVTPDUs are subject to variable transit times between Talker and Listener. The AVTP Presentation Time is used to account for this variability and facilitate synchronization of each AVTPDU's payload data at the Listener.

4.3.2 AVTP Presentation Time

The AVTP Presentation Time represents the gPTP time at which a designated media sample or event within the AVTPDU payload is transferred to the time-sensitive application within each Listener. This enables multiple Listeners to present data at the same time, regardless of the location of the Listeners in the network. It tells a Listener when to start processing (e.g., playing) the stream's data and is also used to recover the stream's media clock.

The AVTP Presentation Time may not be valid in every AVTPDU. If an AVTPDU contains a valid timestamp, then the **tv** (avtp_timestamp valid) bit shall be set to one (1).

4.3.3 AVTP Presentation Time-measurement points

This subclause defines the time-measurement points that are referenced in the calculation of the AVTP Presentation Time.

The AVTP Presentation Time is shown as the Presentation Time Reference Plane in Figure 6.

Copyright © 2016 IEEE. All rights reserved.

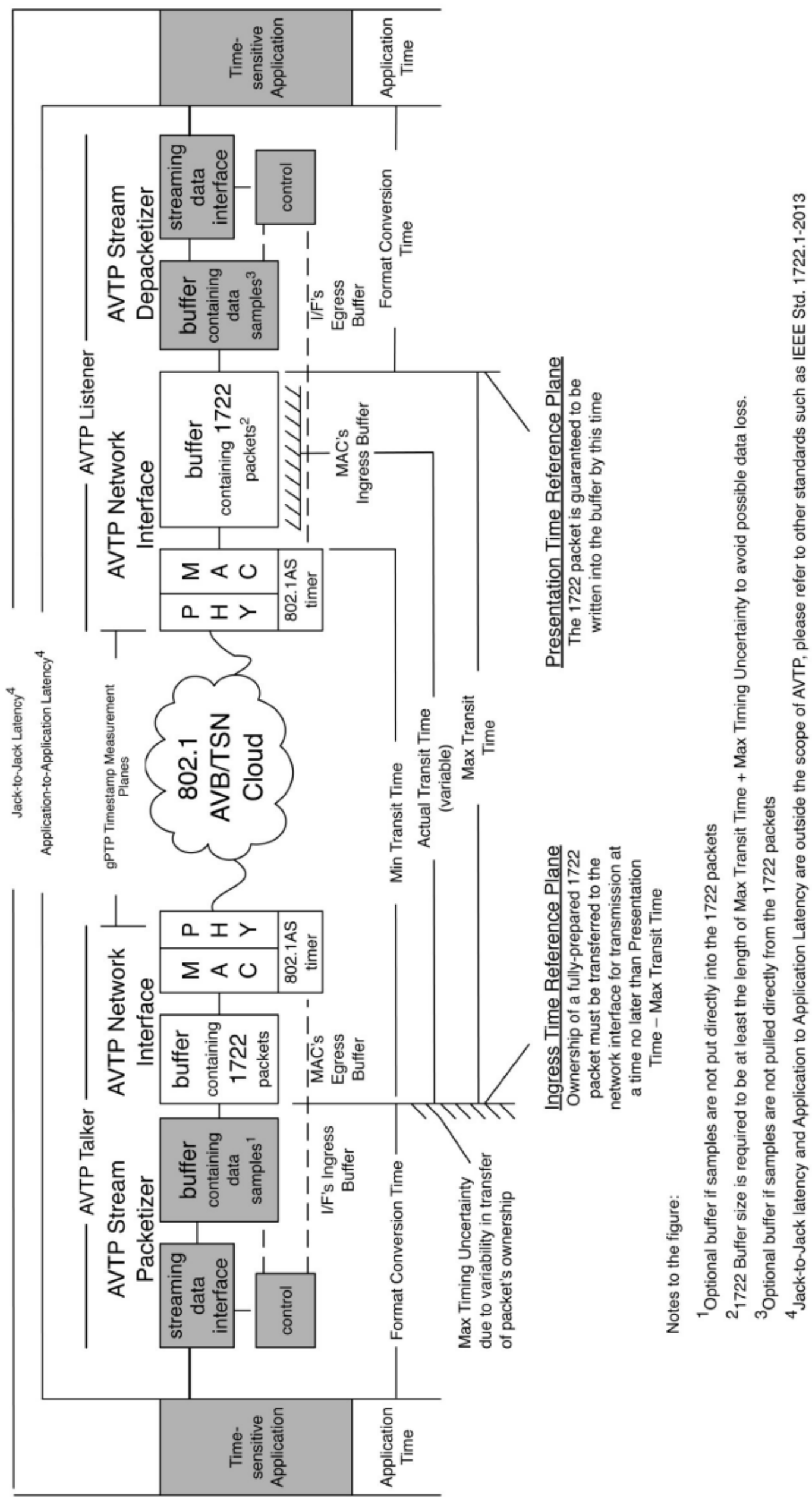


Figure 6—AVTP Timing Reference Planes

Copyright © 2016 IEEE. All rights reserved.

The Max Transit Time is the time between the Ingress Time Reference Plane and the Presentation Time Reference Plane. The Max Transit Time is a function of the speed, size (number of bridges), and configuration of the audio/video bridging (AVB) network.

The default Max Transit Times for SR classes A and B are given in Table 3 (see subclause 6.5 of IEEE Std 802.1BA-2011).

Table 3—AVTP default Max Transit Time

SR class	Max Transit Time (ms)
A	2.0
B	50.0

It is possible to determine an alternate Max Transit Time value for use in a given stream. Guidelines for determining alternate Max Transit Time values are given in 4.3.4. However, the actual mechanism for calculating and assigning this value is outside the scope of this standard.

In most cases, the AVTP Presentation Time corresponds to the first sample in the AVTPDU. The Talker sets the AVTP Presentation Time such that the time at which it transfers ownership of the AVTPDU to the network interface on the Talker (Ingress Time Reference Plane in Figure 6) is at or before the AVTP Timestamp minus the Max Transit Time and within the limits of the Max Timing Uncertainty (Equation (1), where $T_{samples} = 0$).

In cases where the AVTP Presentation Time corresponds to a sample that is not the first sample in the AVTPDU, the packet must be transferred to the network interface at the AVTP Presentation Time minus the Max Transit Time, minus the number of samples before the AVTP Presentation Time in the AVTPDU (Equation (1), where $T_{samples} \neq 0$).

$$PT - TT_{max} - TU_{max} - T_{samples} < T_T < PT - TT_{max} - T_{samples} \quad (1)$$

where

T_T AVTPDU transfer time requirement

PT AVTP Presentation Time

TT_{max} Max Transit Time

TU_{max} Max Timing Uncertainty

$T_{samples}$ Time required to buffer all samples before the sample associated with the AVTP Presentation Time

The Talker shall transfer ownership of stream AVTPDUs to the MAC layer within the constraints of T_T, as defined in Equation (1). If the Talker fails to transfer a stream AVTPDU before T_T, then the AVTPDU may not arrive at the Listener before it is required for audio/video playback. This may result in loss of portions of the media data stream. In any design, there is uncertainty about when this Ingress Time Reference Plane crossing will occur. It is the Talker's responsibility to know the Timing Uncertainty of its own design and to begin the transfer of ownership such that, in the worst case, the crossing of the Ingress Time Reference Plane will occur within the constraints given in Equation (1).

Timing Uncertainty is the potential delay introduced by the transfer of ownership of the frame from the AVTP layer to the media interface. The Max Timing Uncertainty is the maximum amount of transfer delay allowed. Max Timing Uncertainty is different for different traffic priorities. Since higher-priority traffic

Copyright © 2016 IEEE. All rights reserved.

may be queued ahead of lower-priority traffic, there will be more uncertainty in the lower-priority traffic. A Talker's Timing Uncertainty is a combination of many factors, including its own local timer resolution and its worst-case response time to a timer event. The Timing Uncertainty of a Talker shall be no more than the Max Timing Uncertainty amount defined in Table 4.

Table 4—AVTP Max Timing Uncertainty

Class	Max Timing Uncertainty (µs)
A	125
B	1000

The required buffering within a Listener is sufficient to hold the stream's events for the amount of time given in Equation (2) for the class of the stream. If the Listener does not have sufficient buffering for a given stream, a buffer overflow could occur, resulting in data loss.

$$T_{buf} \geq TT_{default} + TU_{max} + PST_{max} \quad (2)$$

where

T_{buf} Listener Buffering Time

$TT_{default}$ Largest of the Default Max Transit Times for the SR classes supported

TU_{max} Max Timing Uncertainty

PST_{max} Max gPTP Skew Time (as defined in subclause B.3 of IEEE Std 802.1AS-2011)

Figure 6 also shows Format Conversion and Reception/Transmission Time on the Talker and Listener. These interfaces are optional media interfaces that could connect to physical devices. These media interfaces are outside the scope of this standard. Any format conversion time required for these interfaces (e.g., buffering time, audio/video codec delay times, etc.) are outside of and in addition to T_{buf} (see Equation (2)).

4.3.4 Calculation of Max Transit Time

In many cases it is desirable to use a Max Transit Time other than the default values listed in Table 3. In the case of an AVTP stream between a Talker and known Listeners it is possible to optimize Max Transit Time to accommodate low-latency applications or to accommodate a large number of intermediate bridges.

The basic method to calculate an appropriate Max Transit Time is to take the worst-case transit time from the Talker to a Listener and choose a Max Transit Time that is greater than or equal to the largest worst-case transit time as shown in Equation (3).

$$\text{MaxTransitTime} \geq \max(\text{WorstCaseTransitTime_1}, \text{WorstCaseTransitTime_2}, \ldots, \text{WorstCaseTransitTime_n}) \quad (3)$$

Factors that affect the calculation of the worst-case transit time include both internal latency in the Talker and Listener and network delays. Network delays are typically related to the physical network topology and number of bridges or other network elements between the Talker to the desired Listeners. Worst-case network delays can be discovered using SRP and the Accumulated Latency as reported within a Talker Advertise attribute.

Once the worst-case transit time has been determined for each Talker/Listener pair, a Max Transit Time for the stream can be determined.

Copyright © 2016 IEEE. All rights reserved.

Figure 7 shows an example of a Talker with two Listener devices on a network, and illustrates the worst-case timing components discussed in this subclause.

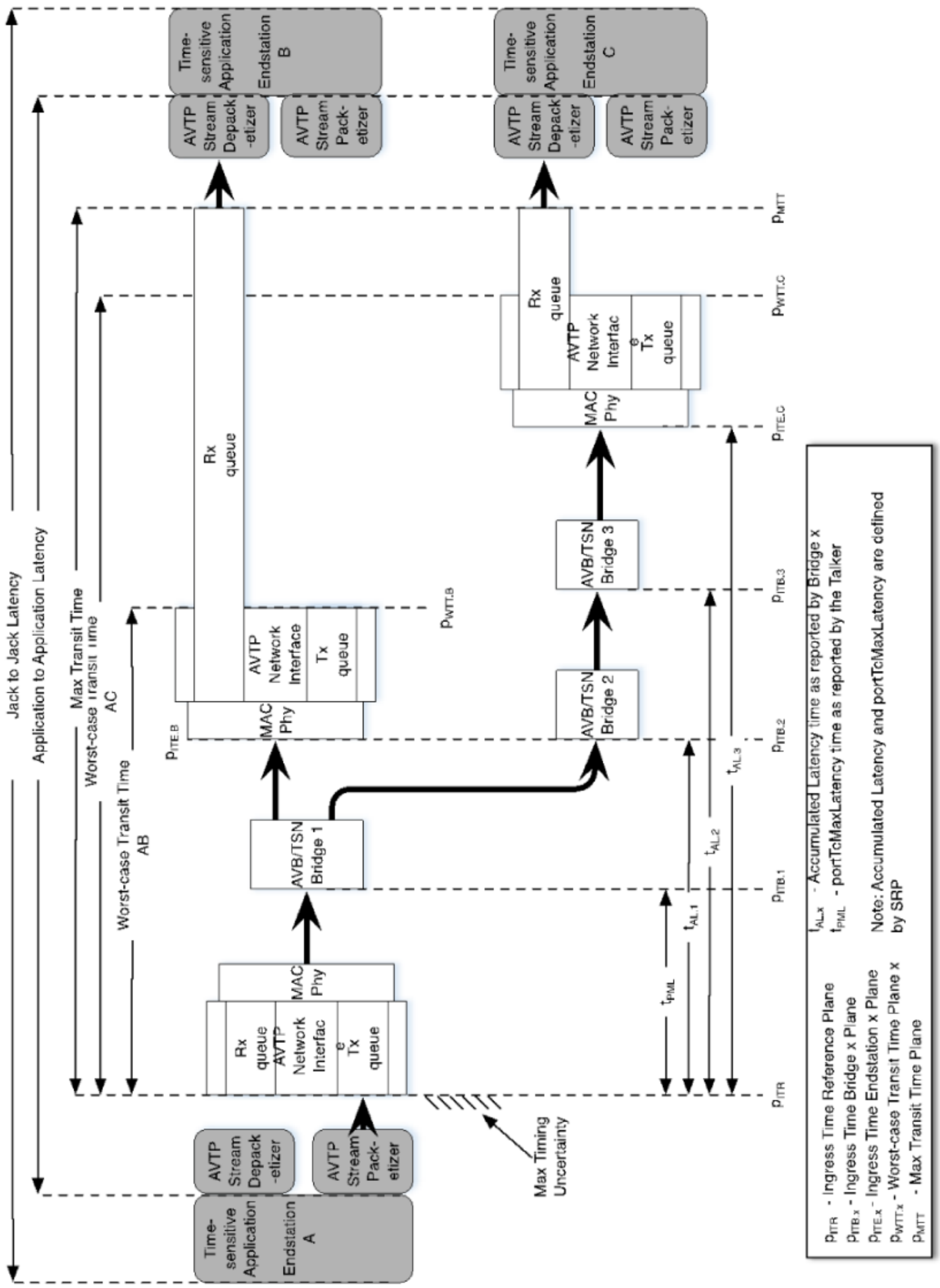


Figure 7—Determining Max Transit Time with multiple Listeners

Copyright © 2016 IEEE. All rights reserved.

Equation (4) shows the calculations for worst-case transit time for an AVTP stream from Talker A to Listener B and Listener C in Figure 7. Equation (5) shows the calculation for the Max Transit Time for an AVTP stream from Talker A to Listener B and Listener C in Figure 7.

$$WTT_{AB} = t_{AL.1} + (P_{WTT.B} - P_{ITE.B})$$
$$WTT_{AC} = t_{AL.3} + (P_{WTT.C} - P_{ITE.C}) \quad (4)$$

$$TT_{max} \geq \max(WTT_{AB}, WTT_{AC}) \quad (5)$$

where

WTT_{AB}	worst-case transit time from Talker A to Listener B
WTT_{AC}	worst-case transit time from Talker A to Listener C
$t_{AL.1}$	Accumulated Latency as reported by SRP for Bridge 1
$t_{AL.3}$	Accumulated Latency as reported by SRP for Bridge 3
$P_{WTT.B}$	Worst-case Timing Plane for Endstation B
$P_{ITE.B}$	Ingress Timing Plane for Endstation B
$P_{WTT.C}$	Worst-case Timing Plane for Endstation C
$P_{ITE.C}$	Ingress Timing Plane for Endstation C
TT_{max}	Max Transit Time

The time $t_{AL.x}$ as reported by SRP includes all queuing and transmission delays for the Talker and Bridges. The reported Accumulated Latency for a Talker is the portTcMaxLatency of the Talker. The reported Accumulated Latency for a Bridge is the worst-case latency from the first bit received by the bridge to the first bit received by the next network device. The reported time is accumulated at each device, so for example, the Accumulated Latency reported by Bridge 3 will include worst-case latency for Talker A, Bridge 1, Bridge 2, and Bridge 3.

The time represented by the equation ($P_{WTT.x} - P_{ITE.x}$) represents the worst-case time from the first bit of the AVTP frame arriving at the Listener until the stream data will be available to the next function in the Listener.

4.3.5 Other timing considerations

There are use cases where it is desirable to synchronize the presentation of media transported in AVTP Streams from multiple Talkers. This can be accomplished by the Talkers subscribing to a single stream that provides the media clocking information for the AVTP Streams that are generated. (See Clock Reference Format in Clause 8.)

This clocking information stream could define an audio media clock, video line or frame clock, or an industrial working clock.

In a system where streams are to be mixed or combined at an endpoint and one or more streams are generated from a recovered media stream, it is required that the difference between the AVTP Presentation Times for the generated stream and the recovered stream be an integer multiple of the nominal period of the stream media clock.

Copyright © 2016 IEEE. All rights reserved.

4.3.6 Protocol layering and other required protocols

This clause documents other required protocols and standards required, but not specified, in this transport-level specification.

All AVTP end stations that transmit or receive time-sensitive streams on a plug-and-play network shall support the following standards:

— Generalized Precision Timing Protocol

— Stream Reservation Protocol

— Forwarding and Queuing Enhancements for Time-Sensitive Streams

4.4 Header formats

4.4.1 Overview

This subclause documents the AVTP header format. The AVTP header includes the basic information such as subtype and version numbers required to decode the AVTPDU. AVTP defines a common header and several specialized header formats.

The AVTPDU common header (4.4.3) contains the basic fields that all formats share, such as the subtype definition and the version of the format.

The AVTPDU common stream header (4.4.4) expands on the AVTPDU common header (4.4.3) and contains the fields that all stream-based formats share. Among other shared fields, this header includes the StreamID for stream identification and a timestamp field for indicating when the included samples are to be presented to the Listener (4.3.2).

The AVTPDU common control header (4.4.5) expands on the AVTPDU common header (4.4.3) and contains the fields that all control formats share. Among other shared fields, this header includes the StreamID for stream identification.

The AVTPDU alternative header (4.4.6) is for formats that do not exhibit the commonalities shared between formats that use the stream or control headers.

4.4.2 AVTP EtherType

The EtherType of all AVTP frames shall be the EtherType given in Table 5.

Table 5—AVTP EtherType

AVTP Ethertype
$22F0_{16}$

Copyright © 2016 IEEE. All rights reserved.

4.4.3 AVTPDU common header format

4.4.3.1 Overview

This subclause documents the first 12 bits of header that are common to all AVTP stream data and control AVTPDUs. These 12 bits consist of the following fields:

— **subtype**: 1 octet

— **h** (header specific): 1 bit

— **version**: 3 bits

Figure 8 shows these fields with offset zero (0) shown as the first octet of the AVTPDU.

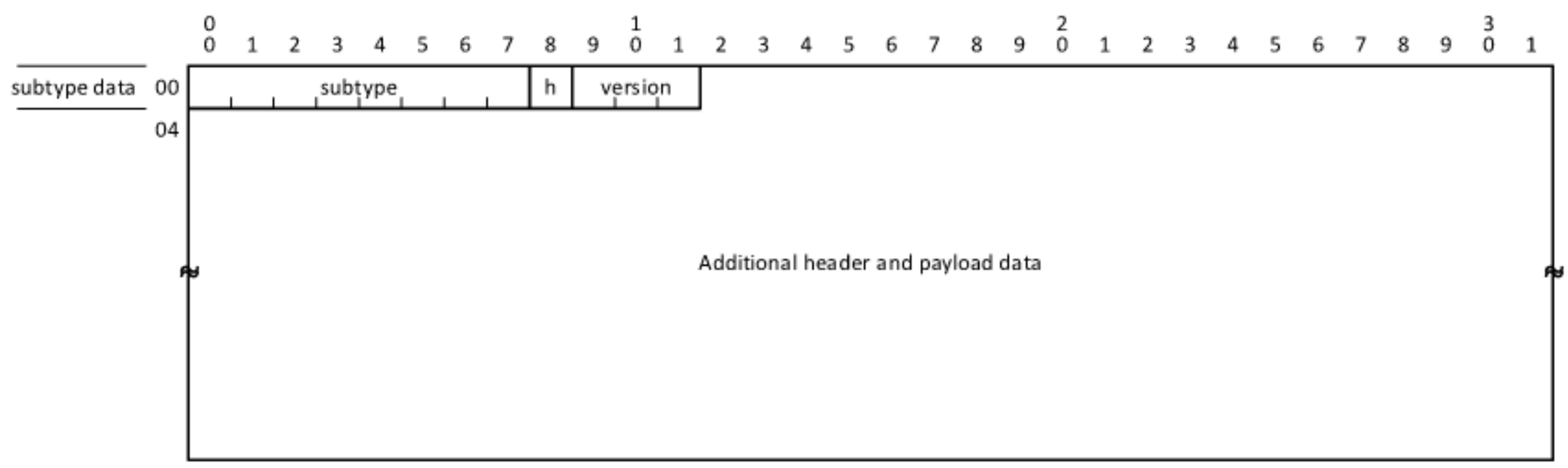


Figure 8—AVTPDU common header fields

4.4.3.2 subtype field

The 1-octet **subtype** field is used to identify the format being carried by AVTP as defined by Table 6. Each format defines its use of AVTP encapsulation. Each subtype is also assigned an *encapsulation style*. Subtypes that are typically composed of a continuous stream of related AVTPDUs are assigned an encapsulation style of *continuous*. Subtypes that are typically composed of discrete AVTPDUs, which have no relationship to each other, are assigned an encapsulation style of *discrete*.

NOTE—The distinction between continuous and discrete stream types is useful in that continuous streams can be extremely high bandwidth and the traffic can overwhelm a network stack. In many cases a special purpose network stack is desired to process continuous-type streams.[1]

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

Copyright © 2016 IEEE. All rights reserved.

Table 6—AVTP stream data subtype values

Value	Name	Description	Reference	Header type	Encapsulation style
00_{16}	61883_IIDC	IEC 61883/IIDC format	Clause 5	Stream (4.4.4)	Continuous
01_{16}	MMA_STREAM	MMA streams	Clause 1	Stream (4.4.4)	Continuous
02_{16}	AAF	AVTP Audio Format	Clause 7	Stream (4.4.4)	Continuous
03_{16}	CVF	Compressed Video Format	Clause 7	Stream (4.4.4)	Continuous
04_{16}	CRF	Clock Reference Format	Clause 8	Alternative (4.4.6)	Continuous
05_{16}	TSCF	Time-Synchronous Control Format	Clause 9.3	Stream (4.4.4)	Continuous
06_{16}	SVF	SDI Video Format	Clause 8	Stream (4.4.4)	Continuous
07_{16}	RVF	Raw Video Format	Clause 12	Stream (4.4.4)	Continuous
08 to $6D_{16}$	Reserved	Reserved	—	—	—
$6E_{16}$	AEF_CONTINUOUS	AES Encrypted Format Continuous	Clause 11	Alternative (4.4.6)	Continuous
$6F_{16}$	VSF_STREAM	Vendor Specific Format Stream	Clause 11	Stream (4.4.4)	Continuous
70 to $7E_{16}$	Reserved	Reserved	—	—	—
$7F_{16}$	EF_STREAM	Experimental Format Stream	Clause 15	Stream (4.4.4)	Continuous
80 to 81_{16}	Reserved	Reserved	—	—	—
82_{16}	NTSCF	Non–Time-Synchronous Control Format	Clause 9.2	Alternative (4.4.6)	Discrete
83 to ED_{16}	Reserved	Reserved	—	—	—
EC_{16}	ESCF	ECC Signed Control Format	Clause 16	Alternative (4.4.6)	Discrete
ED_{16}	EECF	ECC Encrypted Control Format	Clause 17	Alternative (4.4.6)	Discrete
EE_{16}	AEF_DISCRETE	AES Encrypted Format Discrete	Clause 11	Alternative (4.4.6)	Discrete
EF to $F9_{16}$	Reserved	Reserved	—	—	—
FA_{16}	ADP	AVDECC Discovery Protocol	IEEE Std 1722.1™-2013 Clause 6	Control (4.4.5)	Discrete
FB_{16}	AECP	AVDECC Enumeration and Control Protocol	IEEE Std 1722.1-2013 Clause 9	Control (4.4.5)	Discrete
FC_{16}	ACMP	AVDECC Connection Management Protocol	IEEE Std 1722.1-2013 Clause 8	Control (4.4.5)	Discrete
FD_{16}	Reserved	Reserved	—	—	—
FE_{16}	MAAP	MAAP Protocol	Annex B	Control (4.4.5)	Discrete
FF_{16}	EF_CONTROL	Experimental Format Control	Clause 15	Control (4.4.5)	Discrete

4.4.3.3 h (header specific) field

The **header_specific_bit** is a 1-bit field whose meaning is defined by the type of header being used (see 4.4.4.2, 4.4.5.2, and 4.4.6).

Copyright © 2016 IEEE. All rights reserved.

4.4.3.4 version field

The **version** field specifies the version of the format. Unless explicitly defined by a format definition, this field shall be set to zero (0) on transmit and verified on receive. If the Listener receives an AVTPDU with a version that it does not support, it shall discard the AVTPDU.

4.4.4 AVTPDU common stream header

4.4.4.1 Overview

This subclause documents common fields that are used in addition to the common header (see 4.4.3) for stream-related formats.

— **mr** (media clock restart): 1 bit
— **f_s_d** (format-specific data): 2 bits
— **tv** (avtp_timestamp valid): 1 bit
— **sequence_num** (sequence number): 1 octet
— **format_specific_data_1**: 7 bits
— **tu** (timestamp uncertain): 1 bit
— **stream_id**: 8 octets
— **avtp_timestamp**: 4 octets
— **format_specific_data_2**: 4 octets
— **stream_data_length**: 2 octets
— **format_specific_data_3**: 2 octets
— **stream_data_payload**: 0 to *n* octets

The **h** (header_specific) bit (4.4.3.3) of the common header is used to define the following:

— **sv** (stream_id valid): 1 bit

These fields are shown in Figure 9.

Copyright © 2016 IEEE. All rights reserved.

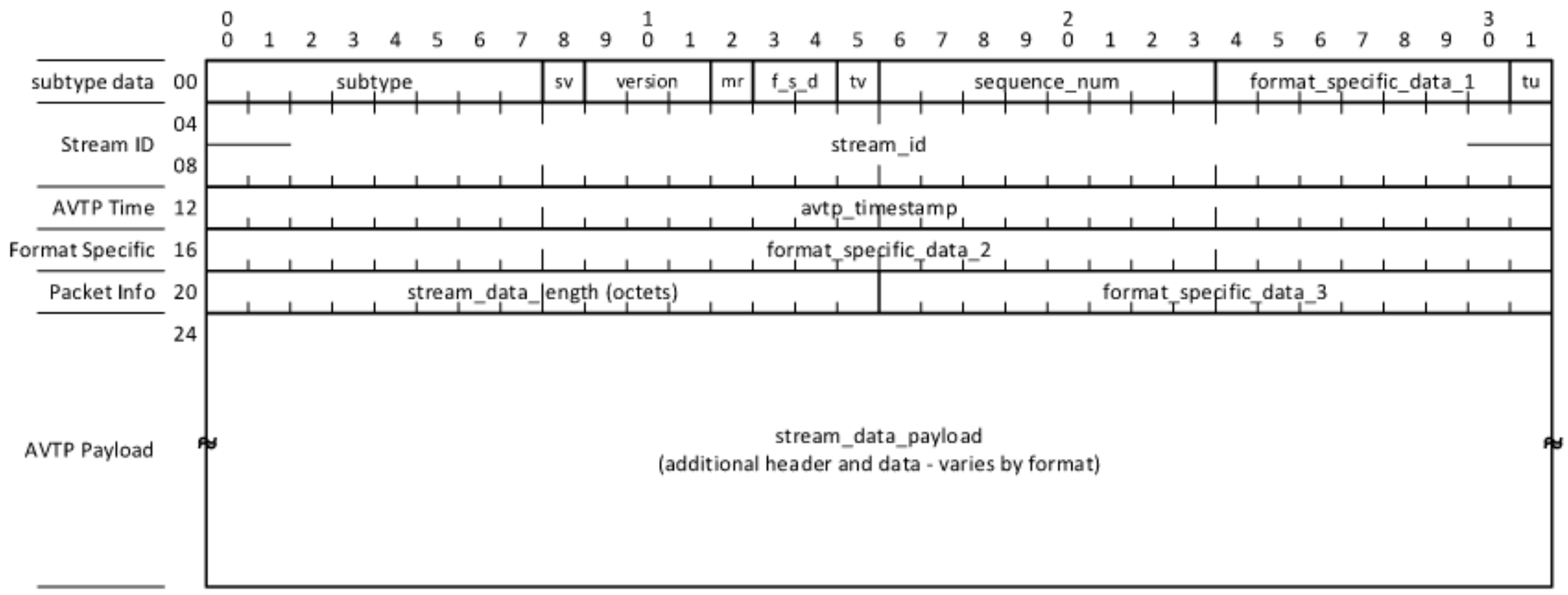


Figure 9—AVTPDU common stream header

4.4.4.2 sv (stream_id valid) field

The **sv** (stream_id valid) field is used to indicate whether the 8-octet **stream_id** field contains a valid StreamID. Since all formats that use the common stream header require a valid StreamID, the value of the **sv** field shall always be one (1).

4.4.4.3 mr (media clock restart) field

The **mr** (media clock restart) field is a 1-bit field used by the Talker to indicate that there has been a change in the source of the media clock. An example of this would be a change from one audio/video input to a different audio/video input where the media clock is supplied from the input (i.e., switching from one Sony/Philips Digital Interconnect Format (S/PDIF) input to a different S/PDIF input). This bit may be used by the Listener to adjust quickly to the new media clock. This kind of change need not be seamless but allows the Listener to take any appropriate action to minimize the disruption to the stream.

This bit is toggled by the Talker each time a media clock restart is needed, and it stays at its new value until a new media clock restart is needed. This toggle helps reduce the likelihood that the media clock restart is missed if it were to appear in a single AVTPDU that just happens to be dropped. Once this bit toggles, it shall remain in its new state for a minimum of eight (8) AVTPDUs for a given continuous stream.

If a Clock Reference Format (CRF) stream (see Clause 8) is being received by a Talker, then any streams deriving timestamps from the CRF stream shall toggle the **mr** bit if a disruption of the CRF stream occurs or if the **mr** bit in the CRF stream has been toggled.

If a Listener is receiving multiple streams that are all derived from a common media clock, the Listener uses only the **mr** bit from the common media clock stream for recovering the media clock.

4.4.4.4 Format-specific data fields

The **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, and **format_specific_data_3** fields are available for use by each given format as specified by the **subtype** field. The specific use of these fields is defined by the format.

Copyright © 2016 IEEE. All rights reserved.

4.4.4.5 tv (avtp_timestamp valid) field

The avtp_timestamp valid (**tv**) field is a 1-bit field used to indicate the validity of the **avtp_timestamp** field time value.

The AVTP Talker shall set the **tv** bit to one (1) if the **avtp_timestamp** field contains a valid value, or zero (0) otherwise.

The AVTP Listener shall ignore the value of the **avtp_timestamp** field if **tv** is set to zero (0).

For how the **avtp_timestamp** field is interpreted and processed, see 4.4.4.9.

4.4.4.6 sequence_num field

The sequence number field (**sequence_num**) is a 1-octet field used to indicate the sequence of AVTPDUs in a stream by the Talker. This field may start at any value for the first AVTPDU of a stream and shall be incremented by one (1), with wrapping, on each subsequent AVTPDU of a stream. This field is to wrap from 11111111_2 to 00000000_2 (FF_{16} to 00_{16}).

The value of this field is determined by the Talker. Listeners can use the sequence number to detect AVTPDUs lost in transit.

A Listener may join a stream at any time after the stream has started, and therefore, a Listener shall be able to tolerate a starting **sequence_num** of any value.

4.4.4.7 tu (timestamp uncertain) field

Although gPTP time is intended to be stable, it is possible for discontinuities to occur in gPTP time. These can be caused by events such as changes in the identity of the gPTP grandmaster clock, changes in the timing source of the grandmaster clock, or other events encountered by a Talker.

When a Talker detects a discontinuity in gPTP time, it should set the **tu** (timestamp uncertain) field to one (1). This indicates to the Listener(s) of the stream that the AVTP Presentation Times contained in the **avtp_timestamp** field may for a limited period of time not correspond to gPTP time. The **tu** bit may be used by Listeners, possibly in conjunction with knowledge of the status of the gPTP clock, to prevent unacceptable disturbances in recovered media streams. Once the Talker has determined that the gPTP clock has returned to a normal state, the **tu** field should be reset to zero (0).

When a Listener detects that the **tu** field has been set to one (1) or detects a discontinuity, either from an gPTP indication or simple observation, it may stop attempting to correlate AVTP Presentation Time to gPTP time until the source of the discontinuity has ceased and the Listener has allowed the appropriate time for the gPTP clock to stabilize.

The time required for the gPTP clock to stabilize is implementation dependent and therefore outside the scope of this standard.

NOTE—When a Listener is in a mode where the **tu** bit is being set, it will usually ignore media clock recovery data and stop adjusting the media clock. Allowing the media clock to free-wheel in this manner will cause the least amount of corruption to the media data as compared to potentially making extreme adjustments to a malfunctioning clock. Timestamps are taken from the media clock like normal. As soon as the **tu** bit is reset, media clock recovery can begin anew and adjust for any clock wander that has occurred since adjustments were suspended.

Copyright © 2016 IEEE. All rights reserved.

4.4.4.8 stream_id field

The **stream_id** field contains the 64-bit StreamID associated with the AVTPDU. The **stream_id** field is used for stream identification.

All stream data AVTPDUs shall contain a valid StreamID in the **stream_id** field, as defined in sublause 35.2.2.8.2 of IEEE Std 802.1Q-2014, and as shown for convenience in Figure 10.

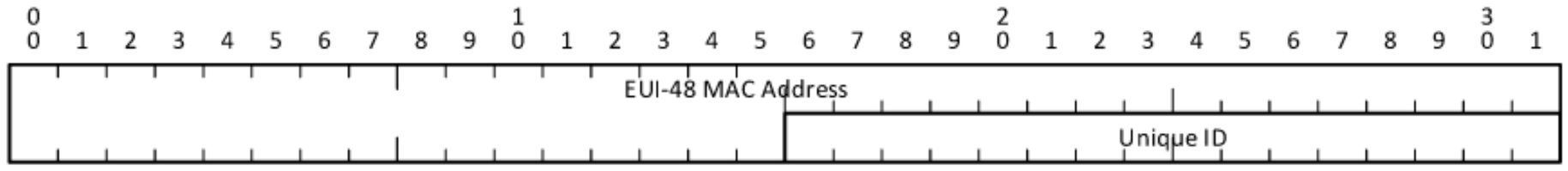


Figure 10—AVTPDU StreamID Format

4.4.4.9 avtp_timestamp field

The 4-octet **avtp_timestamp** field contains the AVTP Presentation Time (see 4.3.2) if the **tv** bit is set to one (1). The **avtp_timestamp** field shall be in units of nanoseconds derived from gPTP time and has a maximum value of $2^{32}-1$ ns. The **avtp_timestamp** field rolls over approximately every 4 s.

The **avtp_timestamp** field is not simply the gPTP nanoseconds field. The **avtp_timestamp** field is given by the formula in Equation (6).

$$\textbf{avtp_timestamp} = (\text{AS_sec} \times 10^9 + \text{AS_ns}) \bmod 2^{32} \tag{6}$$

where

AS_sec the gPTP seconds field

AS_ns value is the gPTP nanoseconds field

If the **tv** bit is zero (0), then the content of the **avtp_timestamp** field is undefined and is ignored.

4.4.4.10 stream_data_length field

The 2-octet **stream_data_length** field contains the length (in octets) of the **stream_data_payload** field.

The maximum value for this field is determined by the MTU and shall be set to allow all stream data AVTPDUs to fit within a single standard frame payload.

4.4.4.11 stream_data_payload field

The **stream_data_payload** contains the data that is associated with or defined by a format. The actual data encapsulation is defined by the format.

Copyright © 2016 IEEE. All rights reserved.

4.4.5 AVTPDU common control header

4.4.5.1 Overview

This subclause documents common fields that are used in addition to the common header (see 4.4.3) for control-related formats.

— **format_specific_data**: 9 bits

— **control_data_length**: 11 bits

— **stream_id**: 8 octets

— **control_data_payload**: 0 to *n* octets

The **h** (header_specific) bit (4.4.3.3) of the common header is used to define the following:

— **sv** (stream_id valid): 1 bit

These fields are shown in Figure 11.

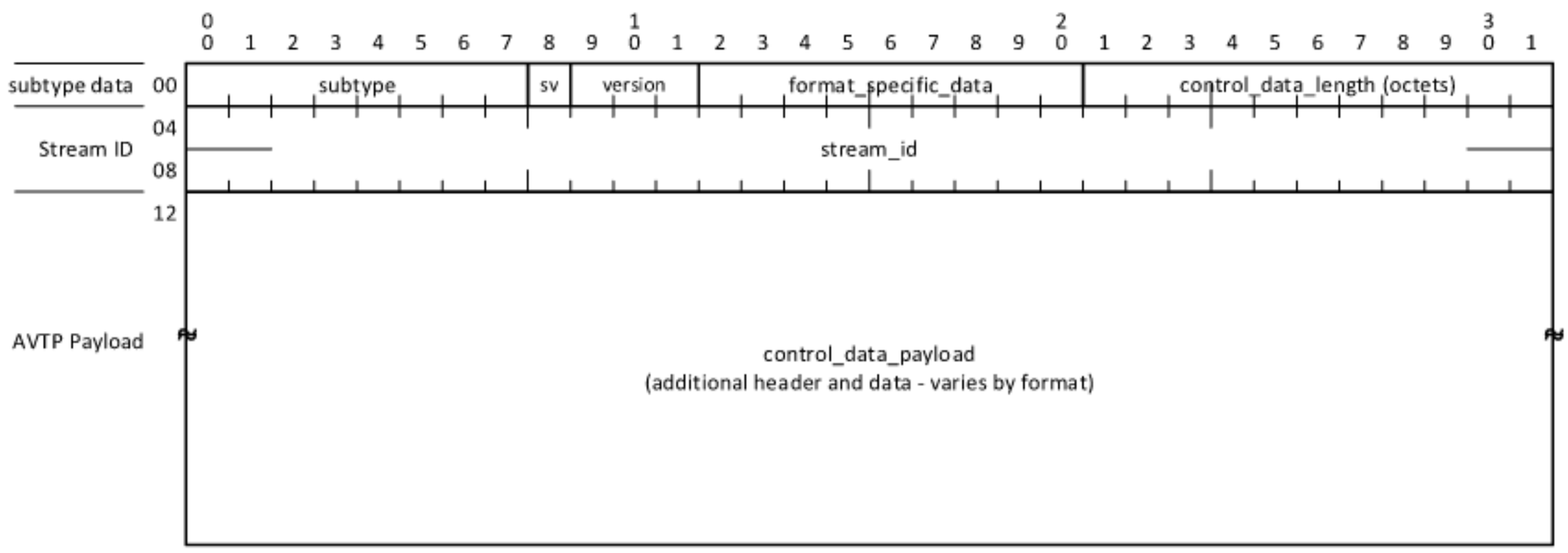


Figure 11—AVTPDU common control header

4.4.5.2 sv (stream_id valid) field

The **sv** (stream_id valid) field is used to indicate whether the **stream_id** field contains a valid StreamID. The options for the value of the **sv** field are defined in Table 7.

Table 7—sv field values

Value	Description
0	The **stream_id** field is not used or is defined by the format
1	The **stream_id** field contains a valid StreamID

4.4.5.3 format_specific_data field

The 9-bit **format_specific_data** field is available for use by the given control format as specified by the **subtype** field. The specific use of this field is defined by the format.

Copyright © 2016 IEEE. All rights reserved.

4.4.5.4 control_data_length field

The 11-bit **control_data_length** field contains the length (in octets) of the **control_data_payload** field.

The maximum value for this field is determined by the maximum transmission unit (MTU) and shall be set to allow all control AVTPDUs to fit in a single standard frame payload.

4.4.5.5 stream_id field

The **stream_id** field is used for stream identification.

If the **sv** field is set to one (1), the **stream_id** field shall contain a valid StreamID, as defined in subclause 35.2.2.8.2 of IEEE Std 802.1Q-2014, and as shown for convenience in Figure 12.

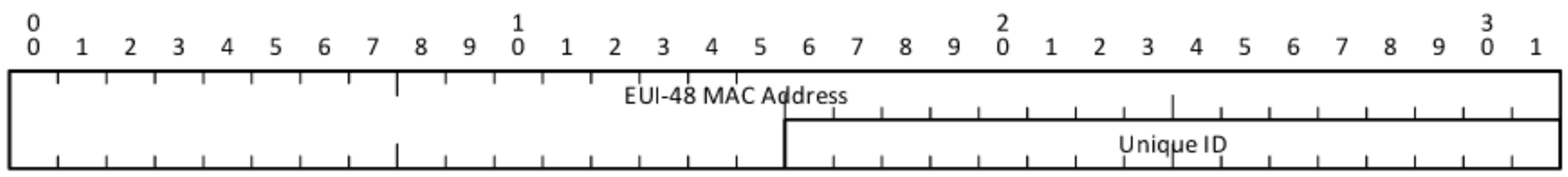


Figure 12—AVTPDU StreamID Format

If the **sv** field is set to zero (0), then the **stream_id** field is reserved unless a specific definition is given by the format.

4.4.5.6 control_data_payload field

The **control_data_payload** contains the data that is associated with or defined by a format. The actual data encapsulation is defined by the format.

4.4.6 AVTPDU alternative header

4.4.6.1 Overview

This subclause documents the fields that are used for formats that do not conform to the common stream or control headers described in 4.4.4 and 4.4.5 respectively.

The usage of the **h** (header_specific) bit (4.4.3.3) of the common header is defined by formats that use the alternative header. If not defined by the format, **h** is reserved.

In addition to the common header (see 4.4.3), one additional field is defined:

— **alternative_data_payload**: 0 to *n* octets

These fields are shown in Figure 13.

Copyright © 2016 IEEE. All rights reserved.

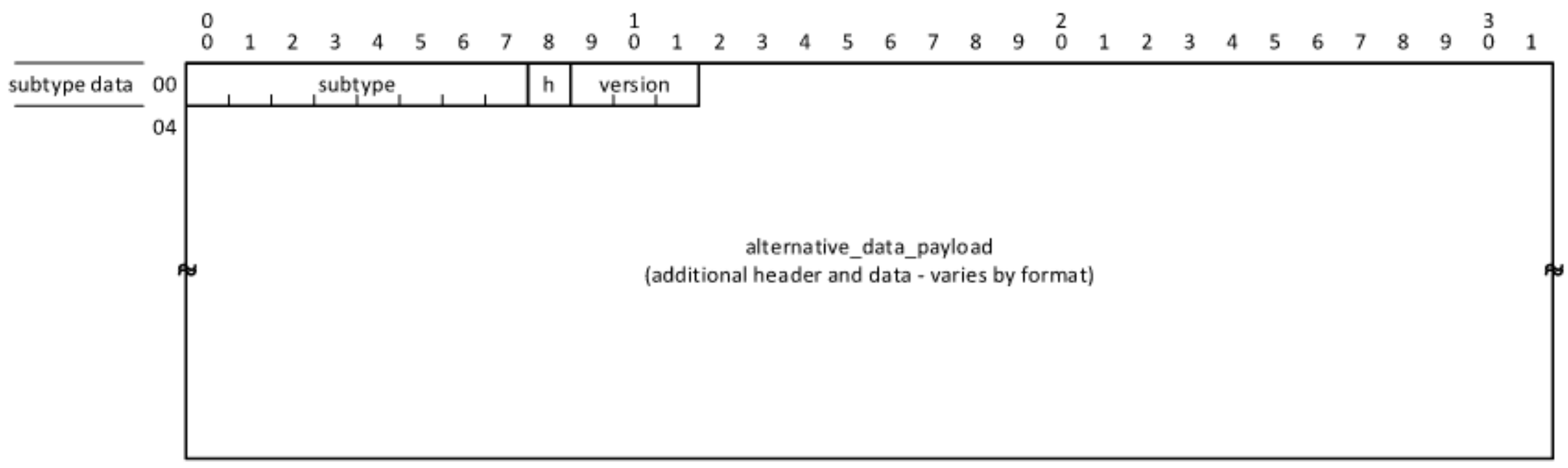


Figure 13—Alternative AVTPDU header

4.4.6.2 alternative_data_payload field

The **alternative_data_payload** field is used by formats to define format-specific headers and, as applicable, a data payload that is used as part of the format. Actual implementation is defined by each format that uses the alternative header.

5. IEC 61883/IIDC Format

5.1 Overview

This clause describes the AVTPDU formats to adapt the IEC 61883 and IIDC protocols to an IEEE 802 network, as well as to adapt the protocols to work with timing and synchronization based on gPTP. This clause does not attempt to document the IEC 61883 protocols, only the changes needed to work in an IEEE 802.1 AVB environment.

AVTP adapts the following IEEE 1394/IEC 61883-type protocols to run in an IEEE 802 environment:

— IEC 61883-4: MPEG2-TS Data Transmission

— IEC 61883-6: Audio and Music Data Transmission Protocol

— IEC 61883-7: Transmission of ITU-R BO.1294 System B

— IEC 61883-8: Transmission of ITU-R BT.601-Style Digital Video Data

— IIDC2 Digital Camera Control Specification Ver. 1.0.0, Jan. 2012

5.2 IEC 61883/IIDC stream data encapsulation

5.2.1 Overview

This encapsulation shall use the common stream header and shall have a **subtype** of 61883_IIDC (see 4.4.3.2).

Copyright © 2016 IEEE. All rights reserved.

This encapsulation uses the following common stream fields (see 4.4.3 and 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

— **stream_id**: 8 octets

— **avtp_timestamp**: 4 octets

— **stream_data_length**: 2 octets

This encapsulation uses the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, **format_specific_data_3**, and **stream_data_payload** fields of the common stream header to define the following format-specific fields:

— **r** (reserved): 1 bit

— **gv** (**gateway_info** valid): 1 bit

— **reserved**: 7 bits

— **gateway_info**: 4 octets

— **tag**: 2 bits

— **channel**: 6 bits

— **tcode**: 4 bits

— **sy**: 4 bits

— **61883_iidc_data_payload**: 0 to *n* octets

These fields are shown in Figure 14.

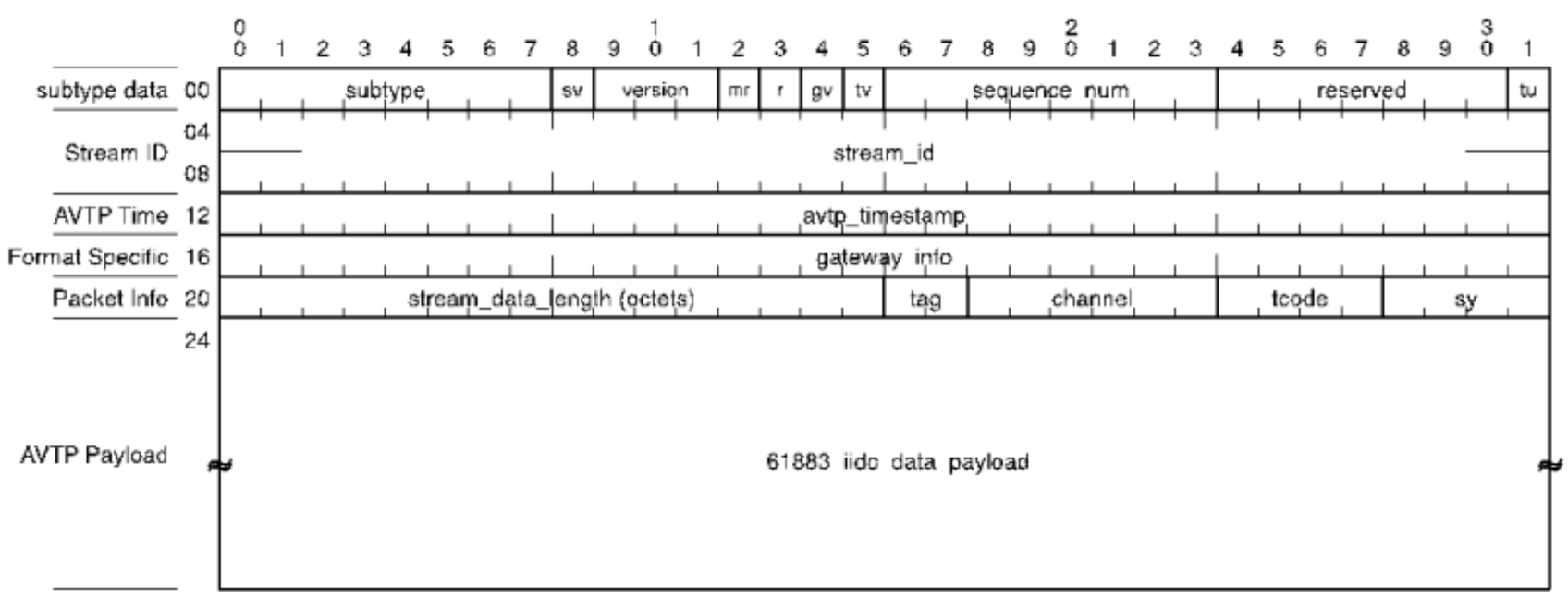


Figure 14—IEC 61883/IIDC common header fields

Copyright © 2016 IEEE. All rights reserved.

5.2.2 gv (gateway_info valid) field

The **gv** (**gateway_info** valid) field is reserved for use by AVTP gateways that are not defined by this standard.

AVTP end stations that do not have an AVTP gateway function shall set this field to zero (0) on transmit and ignore this field on receive.

5.2.3 gateway_info field

The 4-octet **gateway_info** field is reserved for use by AVTP gateways that are not defined by this standard.

AVTP end stations that do not have an AVTP gateway function shall set this field to zero (0) on transmit and ignore this field on receive.

5.2.4 tag field

The 2-bit **tag** field follows the same meaning, format, and rules as specified by IEEE Std 1394. Of the four possible combinations for this field, the following values are supported or not supported as specified in this clause.

Supported by AVTP:

— 00_2: No common isochronous packet (CIP) header included (used by Instrumentation and Industrial Digital Camera (IIDC) 1394 trade association specification) (see 5.3)

— 01_2: CIP header included (see 5.4.3.)

Not supported by AVTP:

— 10_2: Reserved by IEEE Std 1394.1™-2004 clock adjustment [B13]

— 11_2: Global asynchronous stream packet format (used in IEEE Std 1394 for serial bus to serial bus bridges)

5.2.5 channel field

The 6-bit **channel** field follows the same meaning, format, and rules as specified by IEEE Std 1394. The following values are supported as specified:

— 0 to 30 and 32 to 63: Originating channel ID from an IEEE 1394 serial bus via IEEE 1394/IEC 61883 to IEEE 1722/IEC 61883 gateway

— 31: Originating source is on AVTP network (native AVTP)

5.2.6 tcode (type code) field

The 4-bit **tcode** field follows the same meaning, format, and rules as specified by IEEE Std 1394. For AVTP, the only value supported shall be a fixed value of 1010_2 (A_{16}, same as IEEE 1394 isochronous packet format) with the following rules for Talkers and Listeners:

— AVTP Talkers shall set this field to A_{16} on transmit

— AVTP Listeners shall ignore this field on receive

Copyright © 2016 IEEE. All rights reserved.

5.2.7 sy field

Use of the 4-bit **sy** field is application specific and therefore beyond the scope of this standard. The following external standards are known to use this field:

— IIDC 1394-based Digital Camera Specification (used for video start of frame indicator)

— Digital Transmission Content Protection (DTCP)[2] (DTCP Specification [B7], DTCP Volume 1 Supplement [B8])

5.2.8 61883_iidc_data_payload field

The **61883_iidc_data_payload** field contains the data that is associated with or defined by this format. The actual data encapsulation is defined by the **tag** field and given in 5.3.

5.3 IIDC encapsulation

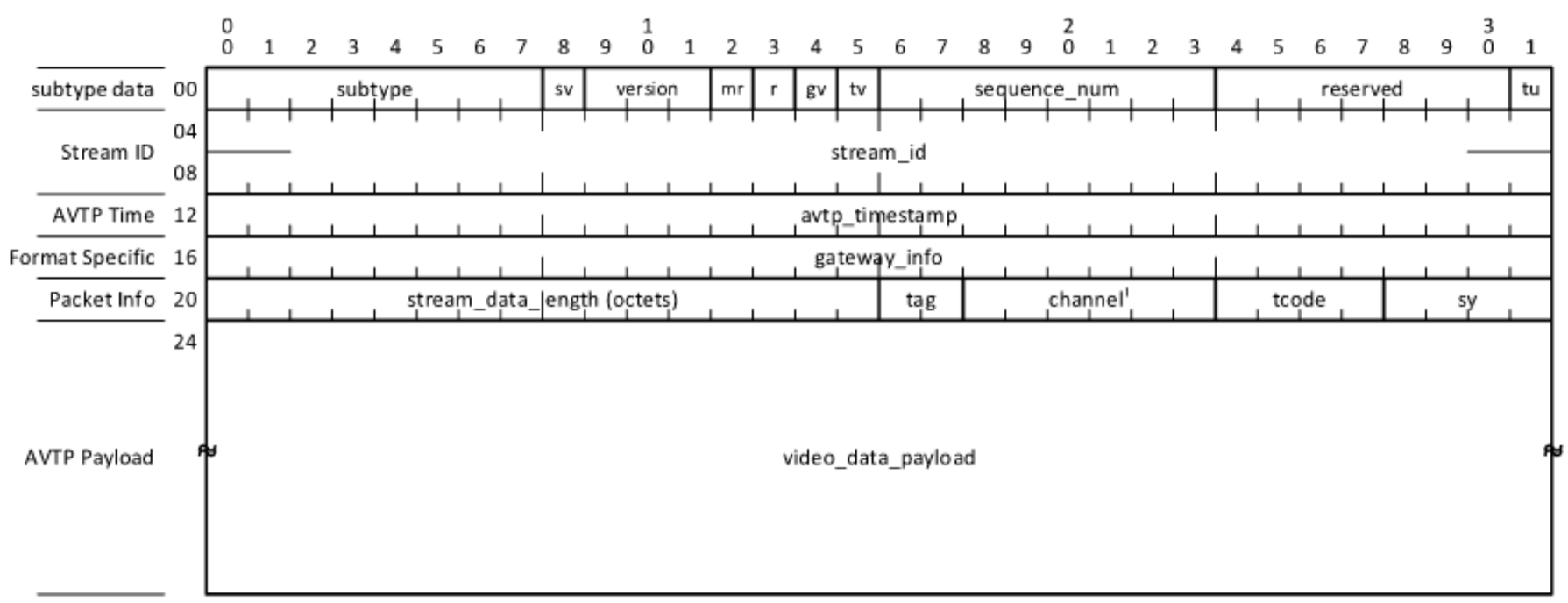
5.3.1 Overview

This subclause describes the AVTPDU format to be used when transporting IIDC data. This option is selected by the **tag** value being set to zero (0).

This encapsulation uses the **61883_iidc_data_payload** field of the IEC 61883/IIDC common header to define the following format-specific field:

— **video_data_payload**: 0 to *n* octets

This is shown in Figure 15.



Notes to figure:

[1]Current standard for IIDC restricts channel ID to 0 - 15

Figure 15—IIDC AVTPDU header fields

[2]Available at http://www.dtcp.com.

Copyright © 2016 IEEE. All rights reserved.

5.3.2 video_data_payload field

5.3.2.1 Overview

The **video_data_payload** field contains the portion of the video frame data being carried by the AVTPDU. The amount of data within the **video_data_payload** field is defined by Table 8. The IIDC Pixel Encoding Format is defined by Table I.1.

Table 8—IIDC video data payload sizes and rates

IIDC Pixel Encoding Format name	Octets per AVTPDU	AVTPDUs per video frame
VGA_MONO8	1280	240
VGA_RGB8_PACKED	1440	640
VGA_YUV422_PACKED	1280	480
VGA_ARGB32	1280	960
VGA_BGRA32	1280	960
XGA_MONO8	1024	768
XGA_RGB8_PACKED	1152	2048
XGA_YUV422_PACKED	1024	1536
XGA_ARGB32	1024	3072
XGA_BGRA32	1024	3072
HD720P_MONO8	1440	720
HD720P_RGB8_PACKED	1440	1920
HD720P_YUV422_PACKED	1440	1280
HD720P_ARGB32	1440	2560
HD720P_BGRA32	1440	2560
SXGA_MONO8	1280	960
SXGA_RGB8_PACKED	1440	2560
SXGA_YUV422_PACKED	1280	1920
SXGA_ARGB32	1280	3840
SXGA_BGRA32	1280	3840
UXGA_MONO8	1280	1500
UXGA_RGB8_PACKED	1440	4000
UXGA_YUV422_PACKED	1280	3000
UXGA_ARGB32	1280	6000
UXGA_BGRA32	1280	6000
HD1080P_MONO8	1440	1440
HD1080P_RGB8_PACKED	1440	4320
HD1080P_YUV422_PACKED	1440	2880
HD1080P_ARGB32	1440	5760
HD1080P_BGRA32	1440	5760
HVGA_MONO8	1280	60
HVGA_RGB8_PACKED	1440	160
HVGA_YUV422_PACKED	1280	120
HVGA_ARGB32	1280	240
HVGA_BGRA32	1280	240
CUSTOM	—	—

The color space names and their refrences back to IIDC2 are defined in Clause 7.3.10 of IEEE Std 1722.1-2013.

Copyright © 2016 IEEE. All rights reserved.

5.3.2.2 MONO8 color space pixel encoding

The MONO8 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.1 and IIDC2 Ver 1.0.0 Section 6.6.3.1.

5.3.2.3 MONO12_PACKED color space pixel encoding

The MONO12_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.2 and IIDC2 Ver 1.0.0 Section 6.6.3.5.

5.3.2.4 MONO16 color space pixel encoding

The MONO16 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.3 and IIDC2 Ver 1.0.0 Section 6.6.3.3.

5.3.2.5 MONO16_SIGNED color space pixel encoding

The MONO16_SIGNED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.4 and IIDC2 Ver 1.0.0 Section 6.6.3.4.

5.3.2.6 YUV411_PACKED color space pixel encoding

The YUV411_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.5 and IIDC2 Ver 1.0.0 Section 6.6.3.2.

5.3.2.7 YUV422_PACKED color space pixel encoding

The YUV422_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.6 and IIDC2 Ver 1.0.0 Section 6.6.3.2.

5.3.2.8 YUV444_PACKED color space pixel encoding

The YUV444_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.7 and IIDC2 Ver 1.0.0 Section 6.6.3.2.

5.3.2.9 RGB_PACKED color space pixel encoding

The RGB_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.8 and IIDC2 Ver 1.0.0 Section 6.6.3.1.

5.3.2.10 RGB16 color space pixel encoding

The RGB16 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.9 and IIDC2 Ver 1.0.0 Section 6.6.3.3.

Copyright © 2016 IEEE. All rights reserved.

5.3.2.11 RGB16_SIGNED color space pixel encoding

The RGB16_SIGNED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.10 and IIDC2 Ver 1.0.0 Section 6.6.3.4.

5.3.2.12 BAYERGR8 color space pixel encoding

The BAYERGR8 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.11 and IIDC2 Ver 1.0.0 Section 6.6.3.1.

5.3.2.13 BAYERGR12_PACKED color space pixel encoding

The BAYERGR12_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.12 and IIDC2 Ver 1.0.0 Section 6.6.3.5.

5.3.2.14 BAYERGR16 color space pixel encoding

The BAYERGR16 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.13 and IIDC2 Ver 1.0.0 Section 6.6.3.3.

5.3.2.15 BAYERRG8 color space pixel encoding

The BAYERRG8 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.14 and IIDC2 Ver 1.0.0 Section 6.6.3.1.

5.3.2.16 BAYERRG12_PACKED color space pixel encoding

The BAYERRG12_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.15 and IIDC2 Ver 1.0.0 Section 6.6.3.5.

5.3.2.17 BAYERRG16 color space pixel encoding

The BAYERGR8 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.16 and IIDC2 Ver 1.0.0 Section 6.6.3.3.

5.3.2.18 BAYERGB8 color space pixel encoding

The BAYERGB8 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.17 and IIDC2 Ver 1.0.0 Section 6.6.3.1.

5.3.2.19 BAYERGB12_PACKED color space pixel encoding

The BAYERGB12_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.18 and IIDC2 Ver 1.0.0 Section 6.6.3.5.

Copyright © 2016 IEEE. All rights reserved.

5.3.2.20 BAYERGB16 color space pixel encoding

The BAYERGB16 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.19 and IIDC2 Ver 1.0.0 Section 6.6.3.3.

5.3.2.21 BAYERBG8 color space pixel encoding

The BAYERGB8 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.20 and IIDC2 Ver 1.0.0 Section 6.6.3.1.

5.3.2.22 BAYERBG12_PACKED color space pixel encoding

The BAYERGB12_PACKED color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.21 and IIDC2 Ver 1.0.0 Section 6.6.3.5.

5.3.2.23 BAYERBG16 color space pixel encoding

The BAYERGB16 color space pixels are encoded as defined in IIDC2 Ver 1.0.0 Section 6.6.2.2 and IIDC2 Ver 1.0.0 Section 6.6.3.3.

5.3.2.24 ARGB32 color space pixel encoding

Each component of the pixel is 8-bits of data and is layed out within the AVTPDU as defined in Figure 16. The data encoding of each component is defined by IIDC2 Ver 1.0.0 Section 6.6.3.1.

0	1	2	3
0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3	4 5 6 7 8 9 0 1
A[K+0]	R[K+0]	G[K+0]	B[K+0]
A[K+1]	R[K+1]	G[K+1]	B[K+1]
A[K+Pn-2]	R[K+Pn-2]	G[K+Pn-2]	B[K+Pn-2]
A[K+Pn-1]	R[K+Pn-1]	G[K+Pn-1]	B[K+Pn-1]

Figure 16—AGRB32 pixel encoding

5.3.2.25 BGRA32 color space pixel encoding

Each component of the pixel is 8-bits of data and is layed out within the AVTPDU as defined in Figure 17. The data encoding of each component is defined by IIDC2 Ver 1.0.0 Section 6.6.3.1.

Copyright © 2016 IEEE. All rights reserved.

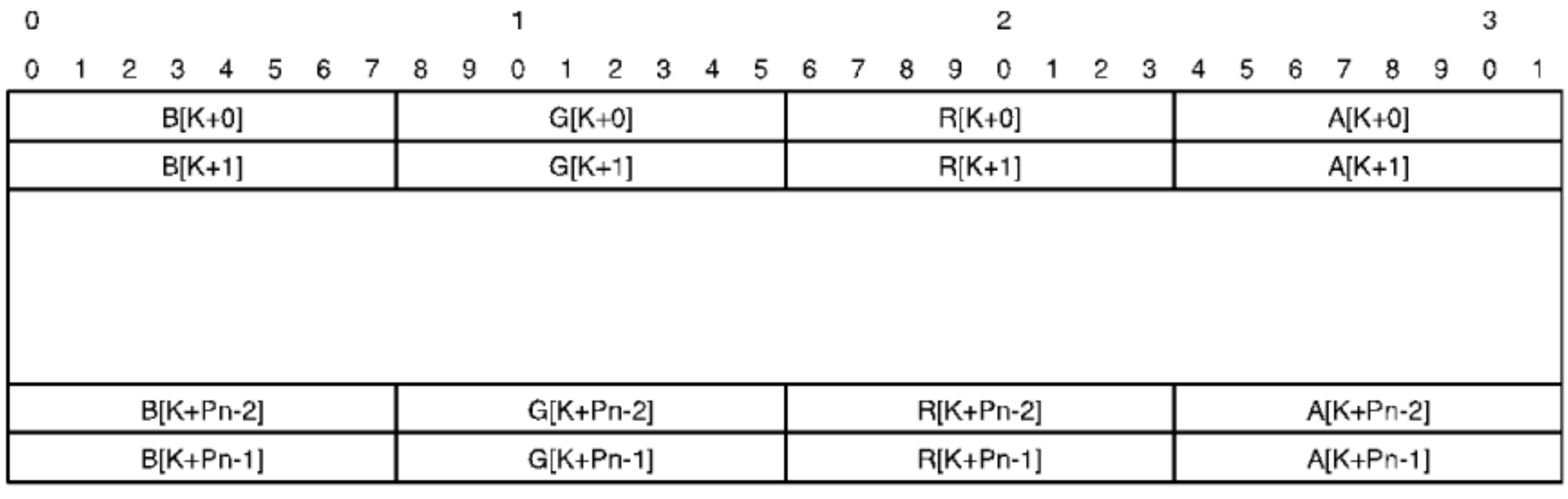


Figure 17 —BGRA32 pixel encoding

5.3.2.26 YUV420_TRI_FULL color space pixel encoding

The YUV420_TRI_FULL color space pixel encoding is a J420 planar representation. Unlike the other pixel encodings YUV420_TRI_FULL groups all of the luminance (Y) values together followed by all of the U chroma values and then all of the V chroma values. As such the entire encoded video frame, as shown in Figure 18 is split into AVTPDUs at the appropriate octet boundaries. The data encoding of each component is defined by IIDC2 Ver 1.0.0 Section 6.6.3.2.

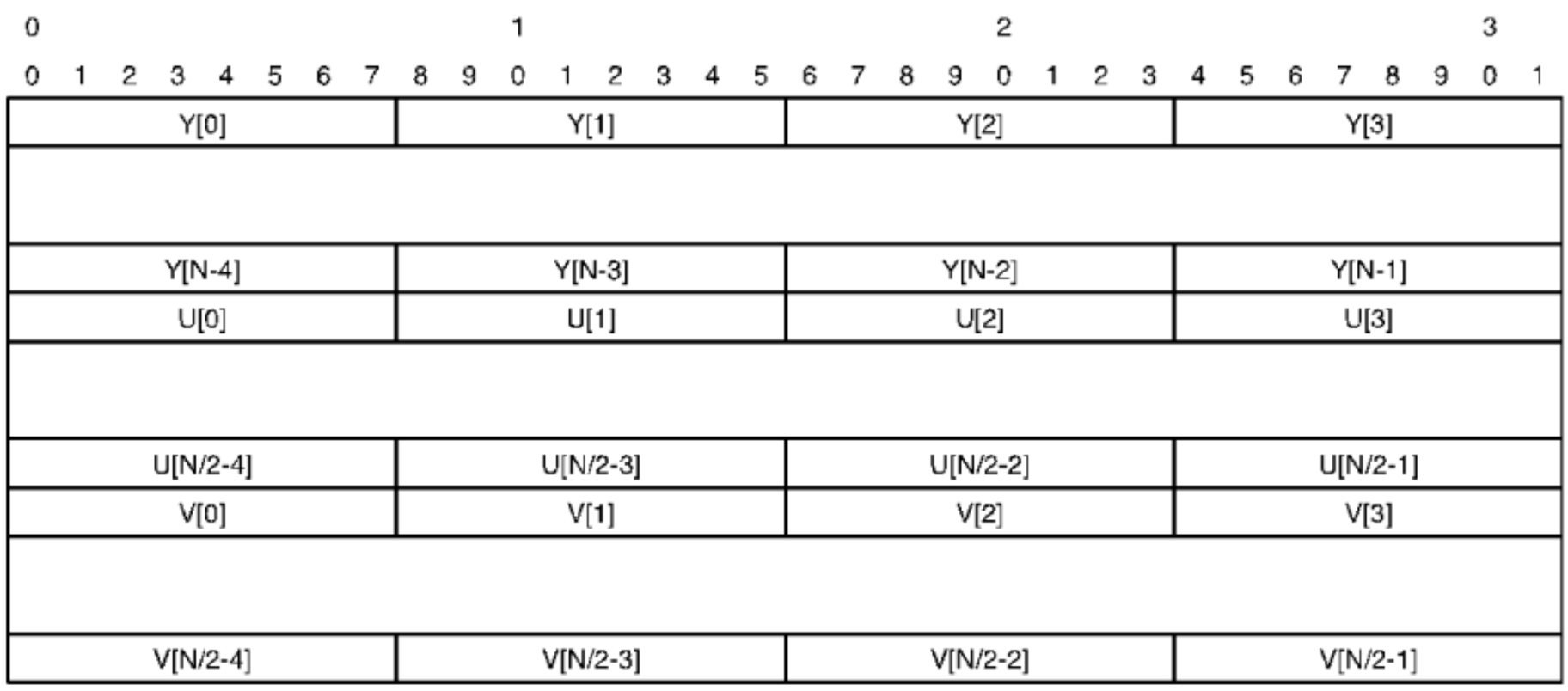


Figure 18 —YUV420_TRI_FULL pixel encoding for whole video frame

5.3.2.27 YUV420_TRI_VIDEO color space pixel encoding

The YUV420_TRI_VIDEO color space pixel encoding is an I420 planar representation. Unlike the other pixel encodings YUV420_TRI_VIDEO groups all of the luminance (Y) values together followed by all of the the U chroma values and then all of the V chroma values. As such the entire encoded video frame, as shown in Figure 19, is split into AVTPDUs at the appropriate octet boundaries. The data encoding of each component is defined by IIDC2 Ver 1.0.0 Section 6.6.3.2.

Copyright © 2016 IEEE. All rights reserved.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Y[0]								Y[1]								Y[2]								Y[3]							
Y[N-4]								Y[N-3]								Y[N-2]								Y[N-1]							
U[0]								U[1]								U[2]								U[3]							
U[N/2-4]								U[N/2-3]								U[N/2-2]								U[N/2-1]							
V[0]								V[1]								V[2]								V[3]							
V[N/2-4]								V[N/2-3]								V[N/2-2]								V[N/2-1]							

Figure 19—YUV420_TRI_VIDEO pixel encoding for whole video frame

5.3.2.28 YUV420_BI_FULL color space pixel encoding

The YUV420_BI_FULL color space pixel encoding is a J420 like biplanar representation. Unlike the other pixel encodings YUV420_BI_FULL groups all of the luminance (Y) values together followed by all of the U chroma and V chroma value pairs. As such the entire encoded video frame, as shown in Figure 18 is split into AVTPDUs at the appropriate octet boundaries. The data encoding of each component is defined by IIDC2 Ver 1.0.0 Section 6.6.3.2.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Y[0]								Y[1]								Y[2]								Y[3]							
Y[N-4]								Y[N-3]								Y[N-2]								Y[N-1]							
U[0]								V[0]								U[1]								V[1]							
U[N/2-2]								V[N/2-2]								U[N/2-1]								V[N/2-1]							

Figure 20—YUV420_BI_FULL pixel encoding for whole video frame

5.3.2.29 YUV420_BI_VIDEO color space pixel encoding

The YUV420_BI_VIDEO color space pixel encoding is a NV12 biplanar representation. Unlike the other pixel encodings YUV420_BI_VIDEO groups all of the luminance (Y) values together followed by all of the the U chroma and V chroma value pairs. As such the entire encoded video frame, as shown in Figure 19, is split into AVTPDUs at the appropriate octet boundaries. The data encoding of each component is defined by IIDC2 Ver 1.0.0 Section 6.6.3.2.

Copyright © 2016 IEEE. All rights reserved.

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Y[0]								Y[1]								Y[2]								Y[3]							
Y[N-4]								Y[N-3]								Y[N-2]								Y[N-1]							
U[0]								V[0]								U[1]								V[1]							
U[N/2-2]								V[N/2-2]								U[N/2-1]								V[N/2-1]							

Figure 21 —YUV420_BI_VIDEO pixel encoding for whole video frame

5.3.3 avtp_timestamp field

The **avtp_timestamp** represents the AVTP Presentation Time (4.3.2) of the IIDC video frame. Every 61883/IIDC AVTPDU of an IIDC video frame shall have the same AVTP Presentation Time.

5.4 IEC 61883 encapsulation

5.4.1 Overview

The following subclauses describe how IEC 61883 formats are adapted for use in AVTP.

5.4.2 Timing and synchronization

5.4.2.1 Overview

This subclause describes how the timing and synchronization methods given in the IEC 61883 group of standards are adjusted to allow their use in AVTP. This clause describes the substitutions that are made to align the terminology given in the IEC 61883 standards to the terminology used in AVTP.

The processing of presentation time, timing, and synchronization for IEC 61883 over AVTP is generally accomplished in the same manner as specified in IEC 61883-1 through IEC 61883-8 over IEEE Std 1394 for processing of presentation time. The main difference between running IEC 61883 protocols over AVTP versus IEEE Std 1394 is the use of gPTP-based time expressed in 4 octets of nanoseconds versus the use of IEEE Std 1394 time expressed in seconds, cycles, and 24.576 MHz cycle offset values.

The usage of the timing and synchronization information included in the CIP header or source packet header (SPH) is generally consistent with the definition in the IEC 61883 series of standards where the main differences are as follows:

— The functionality of the **SYT** field is replaced by the **avtp_timestamp** field.

— The SPH is expressed in gPTP-based nanoseconds.

— For protocols of IEC 61883-4 and IEC 61883-7 where the **SPH** field is set to one (1), the 4-octet source packet header field for each source packet contains the associated AVTP Presentation Time

Copyright © 2016 IEEE. All rights reserved.

of the source packet expressed in gPTP-based nanoseconds instead of IEEE 1394-based seconds, cycle, and cycle offsets.

— In the IEC 61883 series of specifications the term *receiver* is used, whereas in this specification, the term *Listener* is used.

— In the IEC 61883 series of specifications, the term *transmitter* is used, whereas in this specification, the term *Talker* is used.

— IEEE Std 1394 on networks faster than 100 megabits per second allows CIP packets larger than can fit in a single standard Ethernet frame. This standard does not allow this. If an application needs to send more data than can fit in a single Ethernet frame, it shall generate multiple correctly formatted CIP packets, each of which can fit in a standard Ethernet frame.

— Some of the IEC 61883 series of specifications sometimes have options that differentiate "professional," "consumer-use," or "cost-sensitive" equipment. This standard does not have these options, and if a device supports a given format, it shall support all mandatory requirements for that format specified in this document.

NOTE 1—Example: This specification documents the **avtp_timestamp** field function. All AVTP Talkers and Listeners support the field and the processing thereof. As a contrast, the **SYT** field from IEEE Std 1394 in IEC 61883 allows different processing based on the class of equipment.

NOTE 2—This standard does not require a Talker to synchronize frame transmission to an interval of 125 μs for SR class A streams nor to 250 μs for SR class B streams. However, using a period other than integer subdivisions of these class measurement intervals (i.e., /1, /2, /3, /4, etc.) will result in extra bandwidth being reserved, which is therefore unavailable for use by other streams (extra bandwidth is usable by best-effort traffic). For example, an SR class A Talker can send a single frame every 125 μs if it sets MaxIntervalFrames to 1 in the SRP reservation, or it can send a packet every 159 μs with MaxIntervalFrames still equal to 1, or every 12.5 μs by setting MaxIntervalFrames to 10, or every 100 μs by setting MaxIntervalFrames to 2. The bandwidth, in octets per second, reserved by a Talker, can be calculated:

$$\frac{(\text{MaxFrameSize} + \text{media dependent frame overhead}) \times \text{MaxIntervalFrames}}{\text{class measurement interval}}$$

5.4.2.2 IEC 61883-4 timing and synchronization

The timing and synchronization of IEC 61883-4 over AVTP uses the same method as used in IEC 61883-4 over IEEE 1394 networks in that the source packet header quadlet always contains a valid timestamp. In IEC 61883-4 over AVTP, this quadlet contains nanosecond gPTP-based time instead of IEEE 1394 seconds, cycle, and cycle offset time.

For AVTPDUs carrying IEC 61883-4 data, the following rules apply:

a) The timing and synchronization of IEC 61883-4 over AVTP uses the same method as used in IEC 61883-4 over IEEE 1394 networks in that the source packet header quadlet always contains a valid timestamp. In IEC 61883-4 over AVTP, this quadlet contains nanosecond gPTP-based time instead of IEEE 1394 seconds, cycle, and cycle offset time.

b) The constant offset referred to by IEC 61883-4:2004 Section 4.3 that is used when calculating a timestamp is Max Transit Time as defined in 4.3.3.

5.4.2.3 IEC 61883-6 timing and synchronization

For AVTPDUs carrying IEC 61883-6 data, the following rules apply:

Copyright © 2016 IEEE. All rights reserved.

a) In AVTP, the TRANSFER_DELAY described in IEC 61883-6:2014 Section 7.3 is the Max Transit Time defined in 4.3.3, and the DEFAULT_TRANSFER_DELAY value is Default Max Transit Time defined in Table 3.

b) In AVTP, re-transmission of AVTPDUs is not permitted, so paragraph 3 of IEC 61883-6:2014 Section 7 is not applicable to AVTP.

c) Clause 8.4 of IEC 61883-6:2014 describes the 24-bit × 4 audio pack. This type of formatted audio data is not supported by AVTP.

5.4.2.4 IEC 61883-7 timing and synchronization

For AVTPDUs carrying IEC 61883-7 data, the following rule applies:

a) The constant offset referred to by IEC 61883-7:2003 Section 5.1.3 that is used when calculating a timestamp is Max Transit Time as defined in 4.3.3.

5.4.2.5 IEC 61883-8 timing and synchronization

For AVTPDUs carrying IEC 61883-8 data, the following rules apply:

a) Each AVTPDU consists of an AVTPDU header, followed by the CIP header, followed by zero or more source packets (compare IEC 61883-8:2013 Section 4.6.1).

b) An AVTP stream (identified by a unique 8-octet StreamID) that is used to transmit data according to this specification shall only transmit a single stream of video per that stream (compare IEC 61883-8:2013 Section 4.6.2.6).

c) When a nonempty AVTPDU is ready to be transmitted, the Talker shall transmit it as soon as possible within the bounds of the stream transmission as defined by FQTSS. Compare to IEC 61883-8:2013 Section 4.7.1.1, where the start of the isochronous cycle is used.

d) The TRANSFER_DELAY value is derived by replacing the IEEE 1394 transmission worst-case delay with the Max Transit Time (4.3.3). On an AVTP network, it is also possible to replace the values for Allowance for Encryption, Allowance for Decryption, and Decision point to transmit the AVTPDU just missed with zero (0) µs in the calculation of TRANSFER_DELAY. The derivation of the TRANSFER_DELAY value is given in Annex F of IEC 61883-8:2013.

e) The Talker shall construct one or more stream data AVTPDUs at the rate specified by the corresponding FQTSS stream class measurement interval. Each AVTPDU shall comply with the format as specified in Figure 23. When more than one source packet is transmitted that results in multiple AVTPDUs (each with its own CIP header), the Talker shall divide those source packets as evenly as possible across the multiple AVTPDUs.

f) The blocking method may be used by a Talker when it is dealing with fixed counts of source packets per its own transmission interval where it then transmits them to the AVTP network using the queuing mechanisms as specified in FQTSS for per-stream shaping. In the case where there is no data to transmit for a given FQTSS transmission cycle, the Talker may either not transmit any AVTPDUs at all, or it may transmit a stream data AVTPDU containing just a CIP header or a special nonempty AVTPDU that has the ND (NO DATA) flag set to one (1) in its **FDF** field and has the same size of dummy data as a nonempty AVTPDU. Compare to IEC 61883-8:2013 Section 4.7.1.3.

g) The TRANSFER_DELAY value is Max Transit Time defined in 4.3.3.

Copyright © 2016 IEEE. All rights reserved.

5.4.3 IEC 61883 CIP header encapsulation

5.4.3.1 Overview

This subclause describes the AVTPDU format to be used when a CIP header is included in the AVTPDU. This option is selected by the **tag** value being set to one (1).

This encapsulation uses the **61883_iidc_data_payload** field of the IEC 61883/IIDC common header to define the following format-specific fields:

— **qi_1** (quadlet indicator): 2 bits

— **SID** (source identifier): 6 bits

— **DBS** (data block size): 1 octet

— **FN** (fraction number): 2 bits

— **QPC** (quadlet padding count): 3 bits

— **SPH** (source packet header): 1 bit

— **DBC** (data block count): 1 octet

— **qi_2** (quadlet indicator): 2 bits

— **FMT** (stream format): 6 bits

— **cip_fmt_specific_data**: 3 octets

— **cip_data_payload**: 1 to *n* octets

These fields are shown in Figure 22.

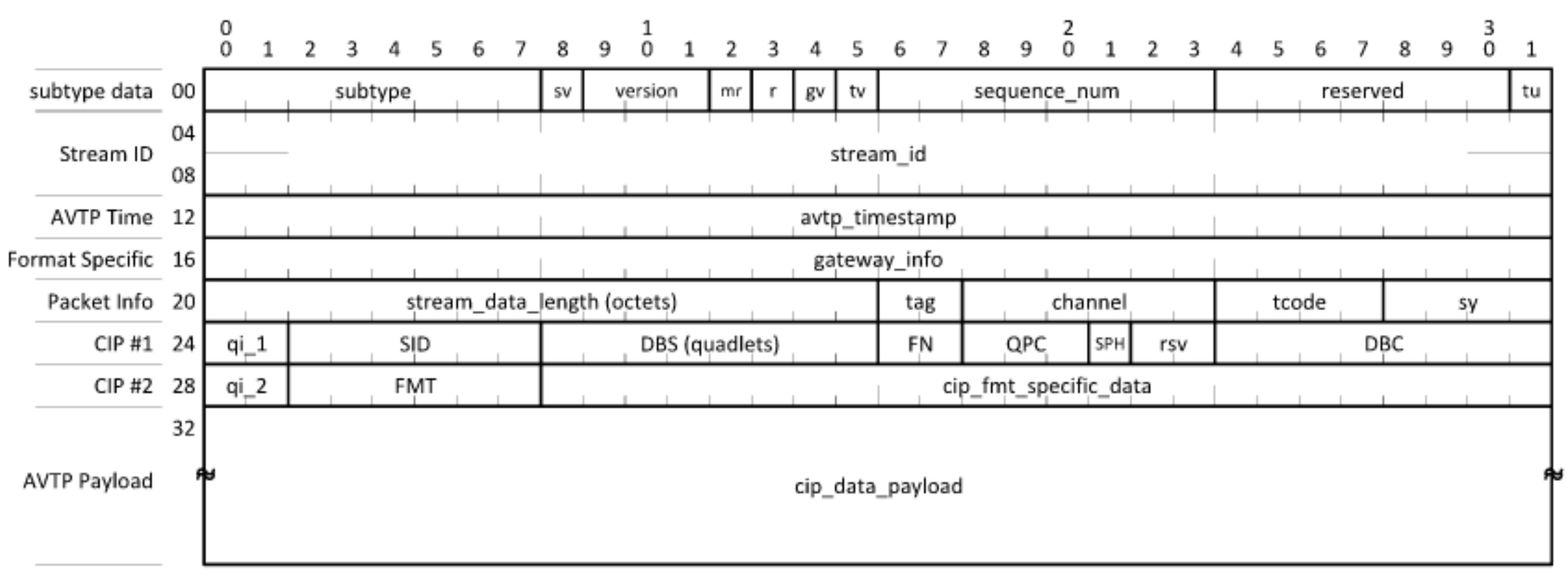


Figure 22—IEC 61883 common header

5.4.3.2 qi_1 (quadlet indicator) field

The 2-bit **qi_1** field has the same definition as IEC 61883-1. AVTP shall only support a fixed value of 00_2 set by Talkers and ignored by Listeners.

Copyright © 2016 IEEE. All rights reserved.

5.4.3.3 SID (source identifier) field

The 6-bit **SID** field has the same definition as IEC 61883-1. For AVTP, it shall use the following values:

— 0 to 62: Originating source ID from an IEEE 1394 serial bus (frame originated from an IEEE 1394/IEC 61883 to IEEE 1722/IEC 61883 interworking unit)

— 63: Originating source is on the AVTP network

5.4.3.4 DBS (data block size) field

The 1-octet **DBS** field has the same definition as IEC 61883-1, which is the size of data blocks in quadlets, as follows:

— 0: 256 quadlets

— 1 to 255: 1 to 255 quadlets

For AVTP, the **DBS** field shall not be set to a value that would cause the AVTPDU length to exceed the MTU of the network. All AVTPDUs of this format shall contain an integral number of data blocks as defined in IEC 61883-1.

5.4.3.5 FN (fraction number) field

The 2-bit **FN** field has the same definition as IEC 61883-1. This is only used in IEC 61883-4 and IEC 61883-7.

5.4.3.6 QPC (quadlet padding count) field

The 3-bit **QPC** field has the same definition as IEC 61883-1. For all types of IEC 61883, this field is always zero (0).

5.4.3.7 SPH (source packet header) field

The **SPH** bit has the same definition as IEC 61883-1.

If set to one (1), the AVTPDU contains IEC 61883-4 or IEC 61883-7 source packets with source packet headers.

If set to zero (0), the AVTPDU does not contain source packets (contains integer number of data blocks).

5.4.3.8 DBC (data block count) field

The 1-octet **DBC** field has the same definition as IEC 61883-1. It contains the sequence number of the first data block in the AVTPDU.

NOTE—See note in 5.4.4.2.

Copyright © 2016 IEEE. All rights reserved.

5.4.3.9 qi_2 (quadlet indicator) field

The 2-bit **qi_2** field has the same definition as IEC 61883-1. AVTP shall only support a fixed value of 10_2 set by Talkers and ignored by Listeners.

5.4.3.10 FMT (stream format) field

The 6-bit **FMT** field has the same definition as IEC 61883-1.

5.4.3.11 cip_fmt_specific_data field

The 3-octet **cip_fmt_specific_data** field is defined differently depending on the value of the **SPH** field (see 5.4.3.7). The use of this field is given in 5.4.4 and 5.4.5.

5.4.3.12 cip_data_payload field

The **cip_data_payload** field is defined differently depending on the value of the **SPH** field (see 5.4.3.7). The use of this field is given in 5.4.4 and 5.4.5.

5.4.4 IEC 61883 (SPH = 0) encapsulation

This subclause describes the AVTPDU format to be used when a CIP header is included in the AVTPDU and no source packet header is included. This option is selected by the **tag** value being set to one (1) and the **SPH** value being set to zero (0), respectively.

Specific uses are given for the following common stream fields:

— **tv (avtp_timestamp** valid): 1 bit

— **avtp_timestamp**: 4 octets

This encapsulation uses the **cip_fmt_specific_data** and **cip_data_payload** fields of the IEC 61883 common header (Figure 22) to define the following format-specific fields:

— **FDF** (format dependent field): 1 octet

— **SYT** (synchronization timing): 2 octets

— **cip_no_sph_payload**: 1 to *n* octets

These fields are shown in Figure 23.

Copyright © 2016 IEEE. All rights reserved.

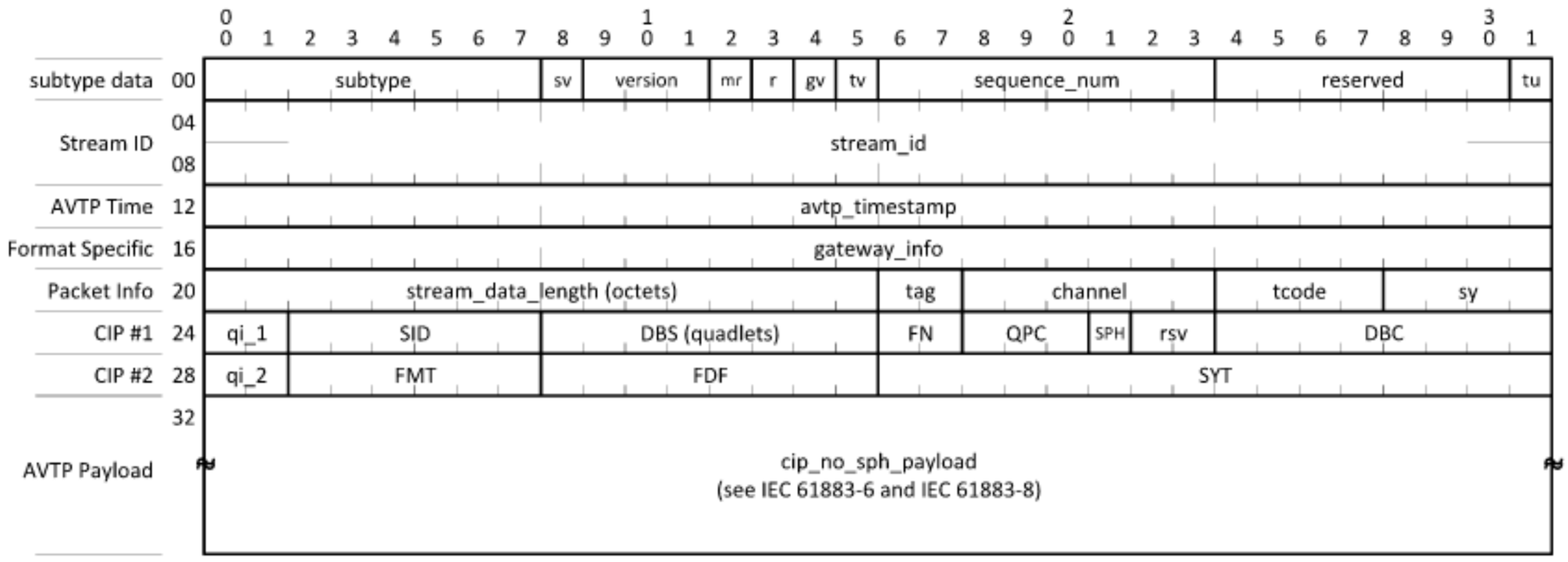


Figure 23—IEC 61883 with no source packet header PDU

5.4.4.1 tv (avtp_timestamp valid) field

The 1-bit **tv** field has the same definition as 4.4.4.5, with additional details given in this subclause.

When transmitting IEC 61883-6 data, the **tv** field shall be set only if the result of Equation (7) refers to a data block that is within the frame.

$$\text{index} = \text{mod}((\text{SYT_INTERVAL} - \text{mod}(\text{DBC}, \text{SYT_INTERVAL})), \text{SYT_INTERVAL}) \quad (7)$$

where

index	the sample offset within the AVTPDU
SYT_INTERVAL	denotes the number of data blocks between two successive valid **avtp_timestamp** fields, which includes one of the data blocks with a valid **avtp_timestamp** field
DBC	the data block count field of a CIP

5.4.4.2 avtp_timestamp field

The 4-octet **avtp_timestamp** field has the same definition as 4.4.4.9, with additional details given in this clause.

When transmitting IEC 61883-6 data, the Talker prepares the **avtp_timestamp** for the data block that meets the condition in Equation (8):

$$\text{mod}(\text{data_block_count}, \text{SYT_INTERVAL}) = 0 \quad (8)$$

where

data_block_count	is the running count of transmitted data blocks
SYT_INTERVAL	denotes the number of data blocks between two successive valid **avtp_timestamp** fields, which includes one of the data blocks with a valid **avtp_timestamp** field. For example, if there are three data blocks between two valid **avtp_timestamp** fields, then the SYT_INTERVAL would be 4.

NOTE—When transmitting IEC 61883-6 data, there are both a field and a variable known as the data block count. The

Copyright © 2016 IEEE. All rights reserved.

field, the **DBC**, is set with the value from the data block count variable when a new AVTPDU is started. Equation (1) of IEC 61883-6:2014 Section 7.2 is run after a data block (e.g., an audio sample frame) has been added to the AVTPDU but before the data block count variable is incremented, and the data block count variable is incremented for every data block that is added to the AVTPDU. When this equation is satisfied then the **tv** field is set to 1 and the timestamp for that data block is set in the **avtp_timestamp** field. If the equation is never satisfied for an AVTPDU then the **tv** field is set to 0 and the **avtp_timestamp** field is ignored.

5.4.4.3 FDF (format dependent field) field

The 1-octet **FDF** field has the same definition as IEC 61883-1.

5.4.4.4 SYT (synchronization timing) field

The 2-octet **SYT** field shall be set to $FFFF_{16}$ by AVTP Talker end stations on transmit, and AVTP Listener end stations shall ignore this field on receive.

5.4.4.5 cip_no_sph_payload field

The **cip_no_sph_payload** field contains audio (IEC 61883-6) or video (IEC 61883-8) samples.

5.4.5 IEC 61883 (SPH = 1) encapsulation

5.4.5.1 Overview

This subclause describes the AVTPDU format to be used when a CIP header is included in the AVTPDU and a source packet header is included. This option is selected by the **tag** value being set to one (1) and the **SPH** value being set to one (1), respectively.

This encapsulation uses the **cip_fmt_specific_data** and **cip_data_payload** fields of the IEC 61883 common header (Figure 22) to define the following format-specific fields:

— **FDF** (format dependent field): 3 octets

— **avtp_source_packet_header_timestamp**: 4 octets

— **cip_with_sph_payload**: 1 to *n* octets

These fields are shown in Figure 24.

Copyright © 2016 IEEE. All rights reserved.

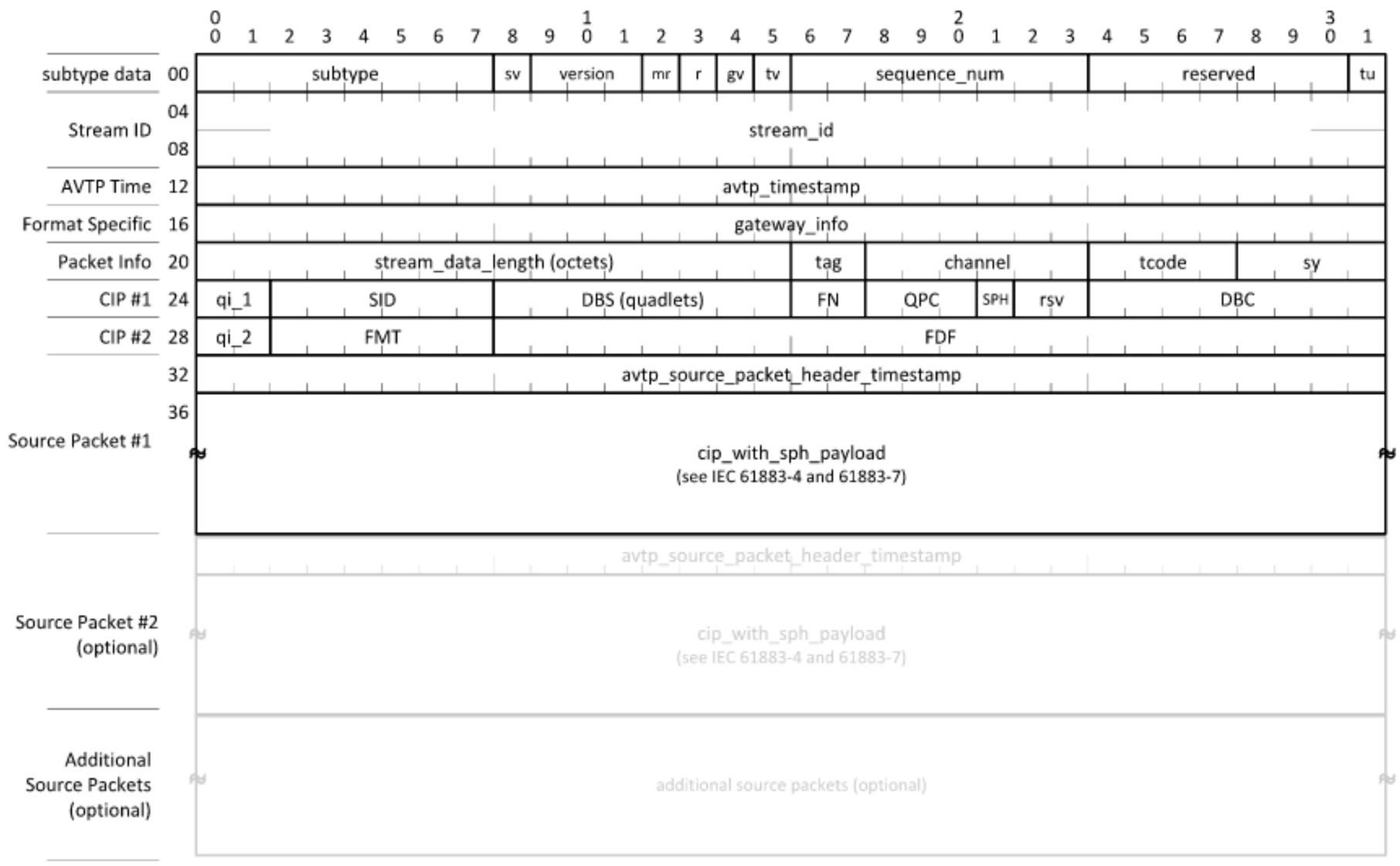


Figure 24—IEC 61883 with source packet header PDU

5.4.5.2 FDF (format dependent field) field

The 3-octet **FDF** field has the same definition as IEC 61883-1.

5.4.5.3 avtp_source_packet_header_timestamp field

Since the **SPH** field is set to a value of one (1), then the stream data AVTPDU contains one or more CIP source packets. Each CIP source packet contains a timestamp as the first quadlet of the source packet followed by an integral number of quadlets as defined by the standard that defines the source packet format.

Formats that set **SPH** to one (1) that are supported by AVTP are IEC 61883-4 and IEC 61883-7.

IEC 61883-4 and IEC 61883-7 define the source packet header quadlet to contain the presentation time of the packet based on IEEE 1394 cycle time. AVTP uses the same formats as defined in IEC 61883-4 and IEC 61883-7, but it uses a gPTP-based presentation time in the same format as defined for the **avtp_timestamp** field (4.4.4.9). For these AVTPDUs, the Talker shall set the **tv** (avtp_timestamp valid) bit to zero (0) for all AVTPDUs (i.e., the AVTP Presentation Time is set in the source packet header instead of in the **avtp_timestamp** field of the AVTPDU common header).

5.4.5.4 cip_with_sph_payload field

The **cip_with_sph_payload** field contains IEC 61883-4 or IEC 61883-7 samples. This field contains all of the source packet following the **avtp_source_packet_header_timestamp**.

Copyright © 2016 IEEE. All rights reserved.

5.4.5.5 Additional source packets

The combination of an **avtp_source_packet_header_timestamp** and a **cip_with_sph_payload** field is designated as a source packet. After the initial source packet, additional source packets may be added to the AVTPDU as needed, while keeping the AVTPDU size smaller than the MTU.

6. MMA Format

AVTPDUs with a subtype value of MMA_STREAM (4.4.3.2) contain the MMA Payload Format. The MMA publishes the MMA Payload Format Specification for AVTP [B24].

This encapsulation shall use the common stream header and shall have a **subtype** of MMA_STREAM (see 4.4.3.2).

7. AVTP Audio Format

7.1 Overview

AVTP Audio Format (AAF) provides a mechanism for transporting audio data over an AVTP network that is optimized for minimum overhead compared to the IEC 61883-6 method (see Clause 5) and maximum utilization of stream reservations. AAF allows single- or multiple-channel audio streams with a variety of sample rates, sample sizes, and sample formats. Within an AAF stream, audio samples are grouped into audio sample frames. Each AAF AVTPDU consists of one or more audio sample frames. For example, a 48 kHz stereo stream with a transmission interval of 125 μs consists of 6 audio sample frames per AAF AVTPDU with 2 samples in each of the audio sample frames.

There are two usable stream data encapsulations defined by AAF: PCM and AES3. Both are derived from the AAF common stream data encapsulation as distinguished by the setting of the **format** field.

7.2 AAF common stream data encapsulation

7.2.1.1 Overview

This encapsulation shall use the common stream header and shall have a **subtype** field of AAF (see 4.4.3.2). This encapsulation uses the following common stream fields (see 4.4.3 and 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

— **stream_id**: 8 octets

Copyright © 2016 IEEE. All rights reserved.

— **avtp_timestamp**: 4 octets

— **stream_data_length**: 2 octets

This encapsulation also uses the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, **format_specific_data_3**, and **stream_data_payload** fields of the common stream header to define the following format-specific fields:

— **rsv** (reserved): 2 bits

— **reserved**: 7 bits

— **format**: 1 octet

— **aaf_format_specific_data_1**: 3 octets

— **rsv** (reserved): 2 bits

— **afsd** (AAF format-specific data): 3 bits

— **sp**: 1 bit

— **evt**: 4 bits

— **aaf_format_specific_data_2**: 1 octet

— **aaf_format_specific_payload**: 0 to *n* octets

These fields are shown in Figure 25.

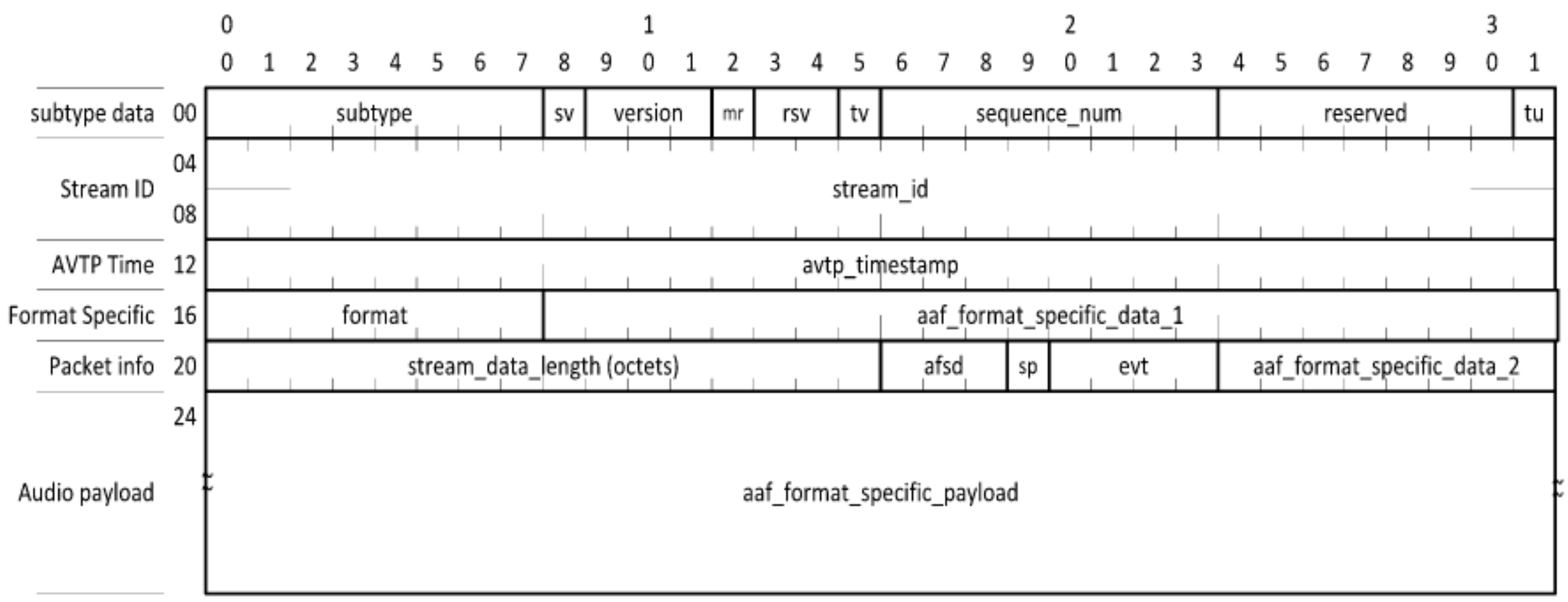


Figure 25—AAF common AVTPDU format

7.2.2 Format-specific data fields

The **aaf_format_specific_data_1**, **aaf_format_specific_data_2**, **afsd**, and **aaf_format_specific_payload** fields are redefined by the PCM or AES3 format, depending on the setting of the **format** field.

Copyright © 2016 IEEE. All rights reserved.

7.2.3 format field

7.2.3.1 Overview

The **format** field describes the format of the samples contained within the stream. The **format** field shall be set to one of the values in Table 9. Table 9 also indicates the AAF AVTPDU format associated with that setting.

NOTE—For additional information on how to configure AAF AES3 streams see Annex K.

Table 9—AAF field values

Value	Name	Description	AAF AVTPDU format
0_{16}	USER	User specified	PCM
1_{16}	FLOAT_32BIT	32-bit floating point. See 7.2.3.2.	PCM
2_{16}	INT_32BIT	32-bit integer	PCM
3_{16}	INT_24BIT	24-bit integer	PCM
4_{16}	INT_16BIT	16-bit integer	PCM
5_{16}	AES3_32BIT	32-bit AES3 format. See 7.4 and Annex K.	AES3
6 to FF_{16}	—	Reserved	N/A

7.2.3.2 Floating point requirements

A **format** field value of FLOAT_32BIT signifies that the audio samples are in the binary32 format described by sublause 3.6 of IEEE Std 754™-2008.

The binary32 values of the payload:

a) Shall not be a NaN (Not a Number), as defined in sublause 3.4a) of IEEE Std 754-2008.

b) Shall not be a negative infinity or positive infinity, as defined in sublause 3.4b) of IEEE Std 754-2008.

c) Shall not be a subnormal number, as defined in sublause 3.4d) of IEEE Std 754-2008.

d) Shall be scaled such that the values of a 0 dBFS sine wave are between −1.0 and +1.0, inclusive.

7.2.4 sp (sparse timestamp) field

AAF provides two timestamping modes, normal and sparse. In normal timestamp mode, the AVTP Presentation Time is contained in every AAF AVTPDU in the stream. In sparse timestamp mode, the AVTP Presentation Time is contained in every eighth AAF AVTPDU. The timestamp mode for a stream is constant throughout the life of the stream.

Because a Listener's clock recovery tuning (i.e., PLL settings) may be different between normal and sparse modes, the **sp** field is used to explicitly define the mode in use for the stream. The **sp** field is also useful for diagnostic purposes to verify that the intended mode is being used correctly.

The **sp** field indicates whether the stream is in sparse timestamp mode. If the **sp** bit is set to one (1) then the stream is in sparse timestamp mode. If the **sp** bit is set to zero (0) then the stream is in normal timestamp mode (Table 10).

Copyright © 2016 IEEE. All rights reserved.

Table 10—sp field description

Value	Description
0	Normal operation, timestamp in every AAF AVTPDU
1	Sparse mode, timestamp in every eighth AAF AVTPDU

See 7.5 for a description of how the **tv** and **avtp_timestamp** fields are populated when the **sp** bit is set.

7.2.5 evt field

The **evt** field contains 4 bits that are reserved for upper-level protocols to indicate events in the stream. The specific use of these bits is outside the scope of this standard.

7.3 AAF PCM stream data encapsulation

7.3.1.1 Overview

When the value of the **format** field is set to a value for which Table 9 indicates the use of the PCM encapsulation, the format of the AAF AVTPDU is given in this clause.

This encapsulation uses the **aaf_format_specific_data_1**, **afsd**, **aaf_format_specific_data_2**, and **aaf_format_specific_payload** fields of the AAF common header (Figure 25) to define the following specific fields:

— **nsr** (nominal sample rate): 4 bits

— **rsv** (reserved): 2 bits

— **channels_per_frame**: 10 bits

— **bit_depth**: 1 octet

— **rsv** (reserved): 3 bits

— **reserved**: 1 octet

— **pcm_data_payload**: 0 to *n* octets

These fields are shown in Figure 26.

Copyright © 2016 IEEE. All rights reserved.

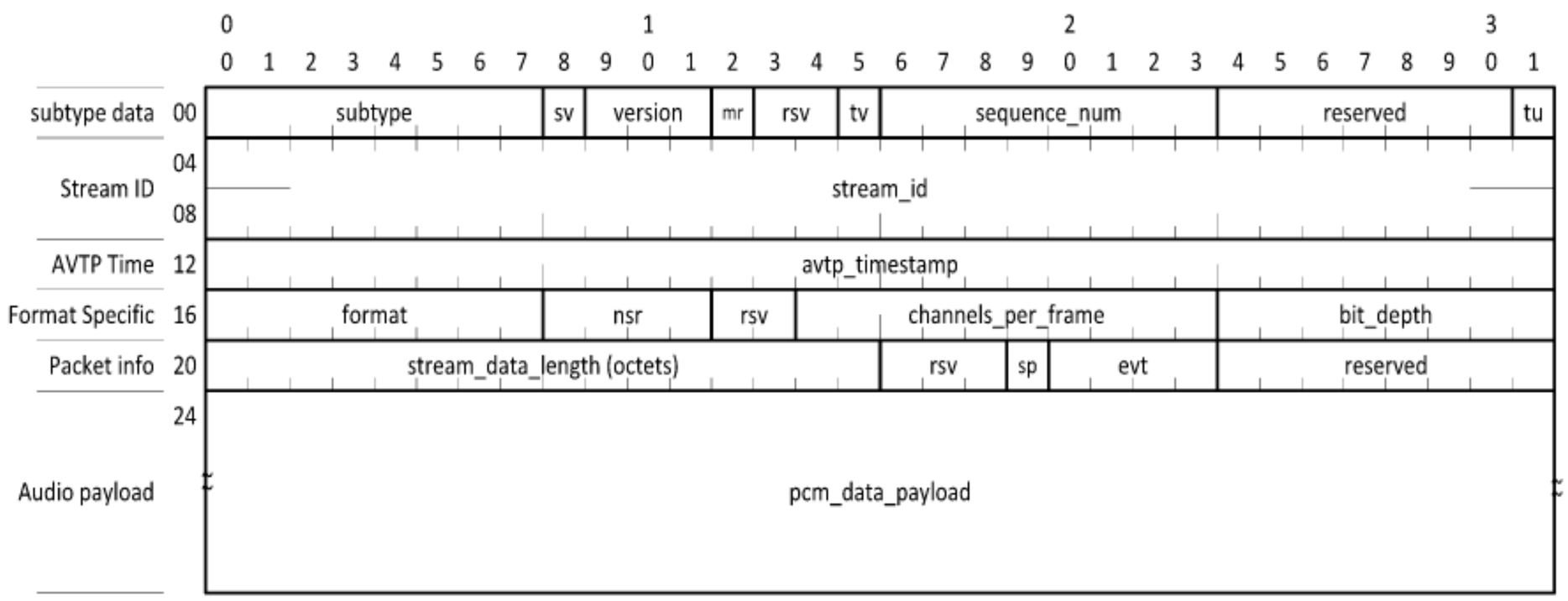


Figure 26—AAF PCM AVTPDU format

7.3.2 nsr (nominal sample rate) field

For **nsr** field describes the sample rate of the PCM audio stream. The **nsr** field shall be set to one of the values in Table 11.

Table 11—nsr field values

Value	Description
0_{16}	User specified
1_{16}	8 kHz
2_{16}	16 kHz
3_{16}	32 kHz
4_{16}	44.1 kHz
5_{16}	48 kHz
6_{16}	88.2 kHz
7_{16}	96 kHz
8_{16}	176.4 kHz
9_{16}	192 kHz
A_{16}	24 kHz
B to F_{16}	Reserved

7.3.3 channels_per_frame field

The **channels_per_frame** field shall be set to the number of audio channels represented in the audio sample frame, a positive integer greater than zero (0). For example, an audio stream conveying "7.1 surround sound" shall be indicated as having 8 channels.

7.3.4 bit_depth field

The **bit_depth** field indicates which bits within the sample word the Talker will use to transmit valid data, and shall not be set to a value of zero (0). The value of **bit_depth** shall not be set to a number which is larger than the size of the format set in the **format** field. If the **format** field is set to a FLOAT_32BIT format then the **bit_depth** field shall be set to 32.

Copyright © 2016 IEEE. All rights reserved.

The bits within the sample used for data start with bit zero (0) and continue to the (**bit_depth** − 1) bit. All unused sample bits shall be set to zero (0).

Figure 27 shows how a 26-bit PCM audio sample from an ADC is mapped onto a 32-bit integer AVTPDU sample; valid sample bits are designated with a "v."

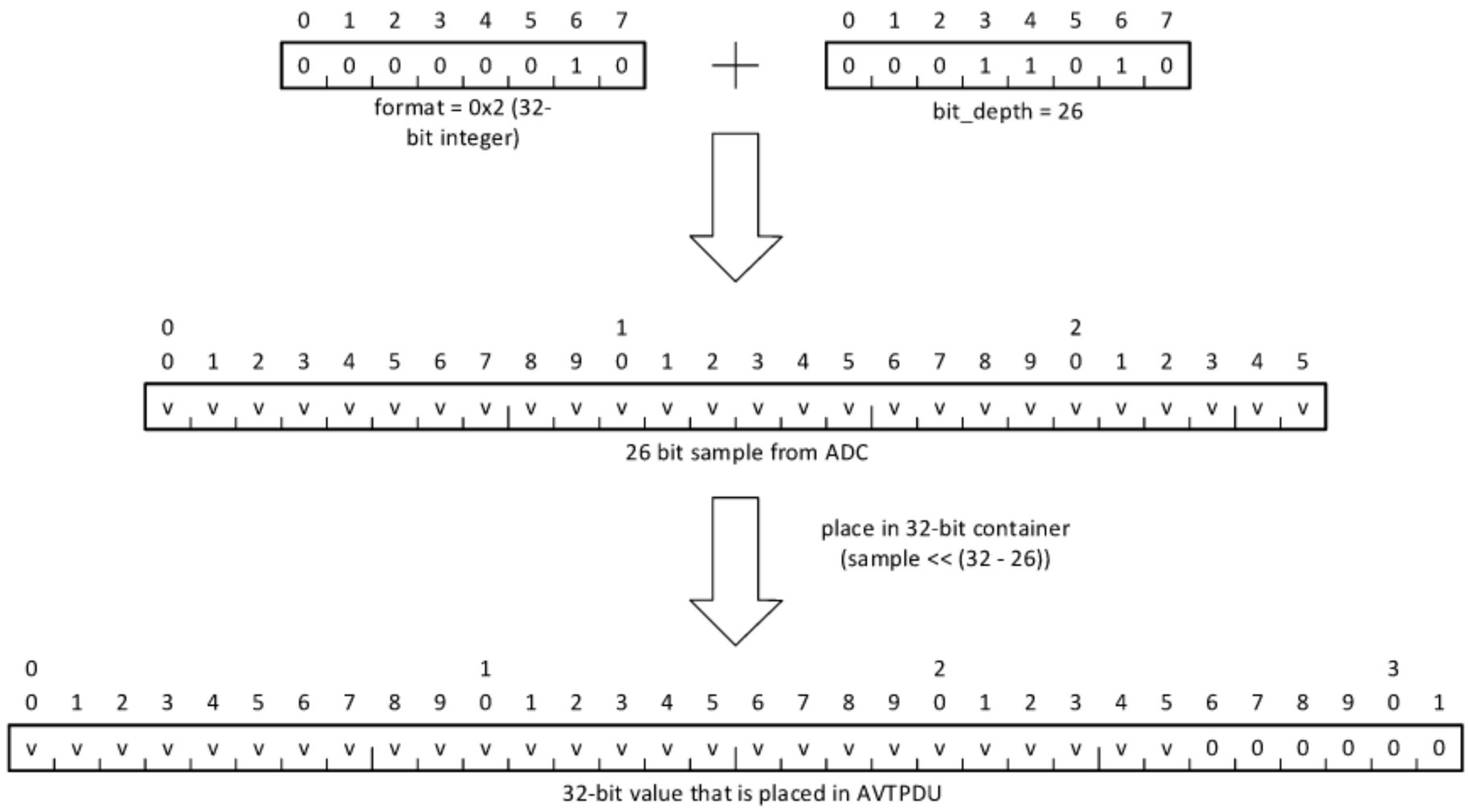


Figure 27—Example of 26-bit ADC sample mapped to PCM AVTPDU 32-bit payload value

NOTE—The determination of what is considered a full-range signal and the method for scaling sample values to allow a particular **bit_depth** to span the dynamic range of a digital to analog converter (DAC) is Listener specific and outside the scope of this standard.

7.3.5 pcm_data_payload field

The **pcm_data_payload** field contains the audio samples. All AAF AVTPDUs within a stream shall contain the same number of samples. The byte order of the samples shall be network byte order (i.e., the most-significant byte of a multi-byte sample is located at the least-numbered AVTPDU byte offset, per 3.4.1).

NOTE—Because all AAF AVTPDUs in a stream are required to have the same number of audio samples, the last AVTPDU in a stream cannot be shorter than previous AAF AVTPDUs. That means that if there are not enough samples to fill an entire AAF AVTPDU then the Talker can either drop (not transmit) the few remaining samples or pad the remainder of the AAF AVTPDU with synthesized samples. The values of such synthesized samples are outside the scope of this standard.

If the **channels_per_frame** field is greater than one then the audio samples are interleaved chronologically into audio sample frames. For example, Figure 28 shows the layout of a 2-channel, 32-bit PCM integer, 6-samples-per-packet **pcm_data_payload** field. Figure 29 shows the layout of an 8-channel, PCM 16-bit integer, 6-samples-per-packet **pcm_data_payload** field.

Copyright © 2016 IEEE. All rights reserved.

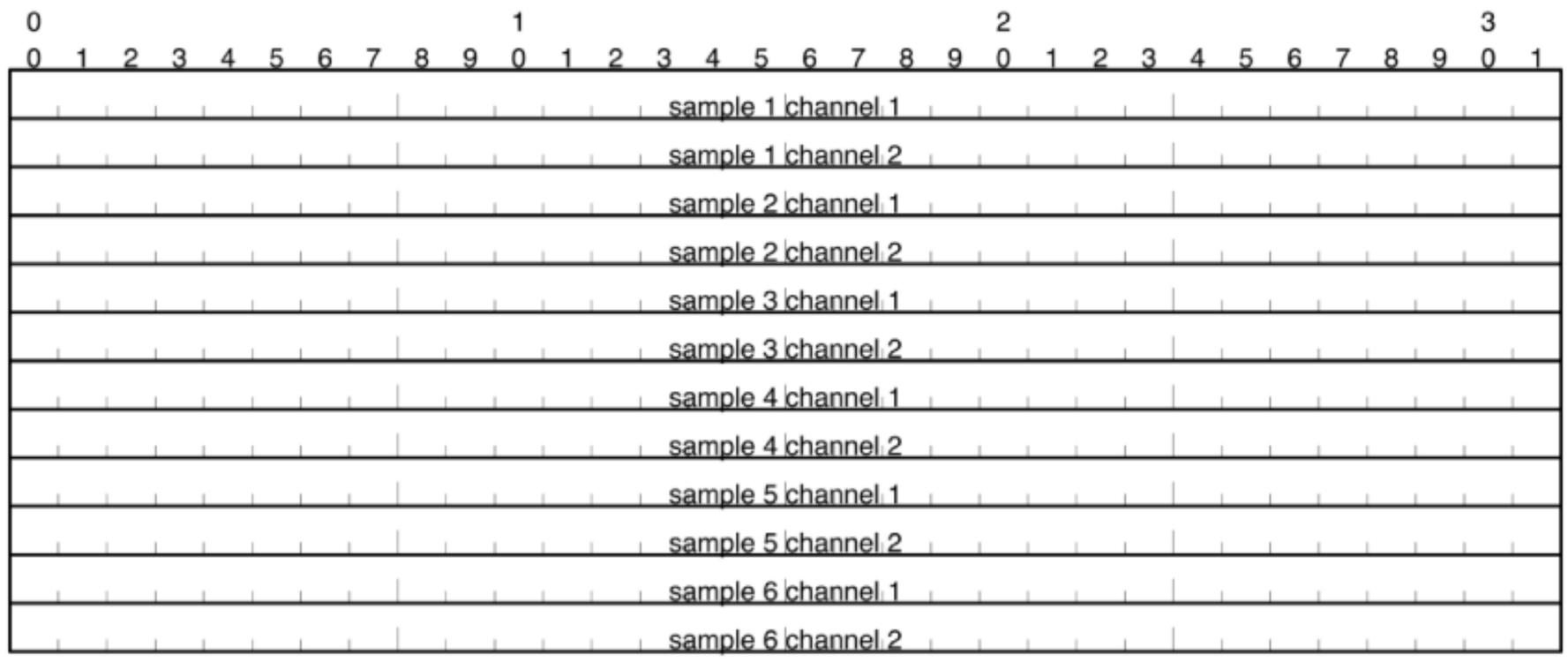


Figure 28—AAF 32-bit PCM integer, 2-channel

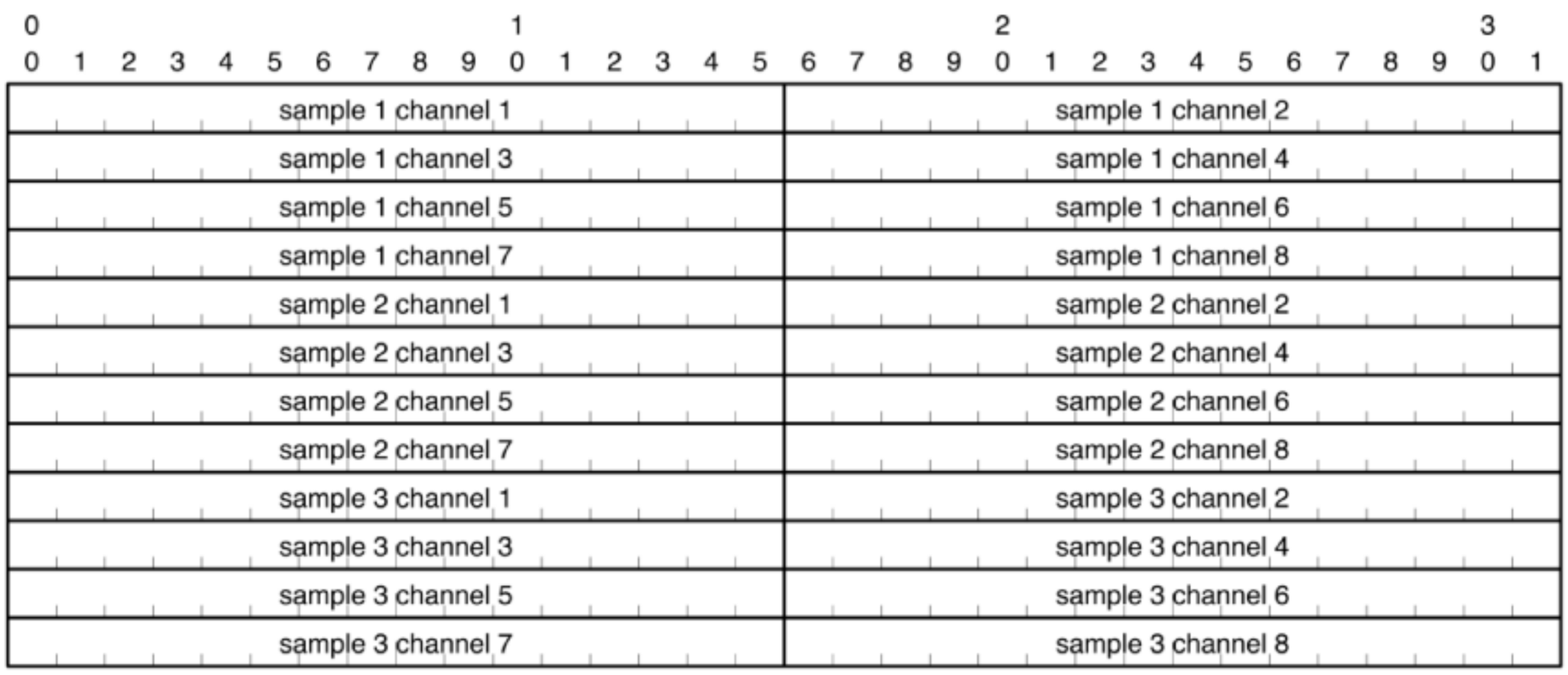


Figure 29—AAF 16-bit PCM integer, 8-channel

All audio samples are packed according to their size specified by the **format** field. A 32-bit float or integer audio sample uses exactly 4 octets of data space. A 24-bit integer audio sample uses exactly 3 octets, and a 16-bit integer audio sample uses exactly 2 octets within the **pcm_data_payload** field, as shown in Figure 30 and Figure 31, respectively.

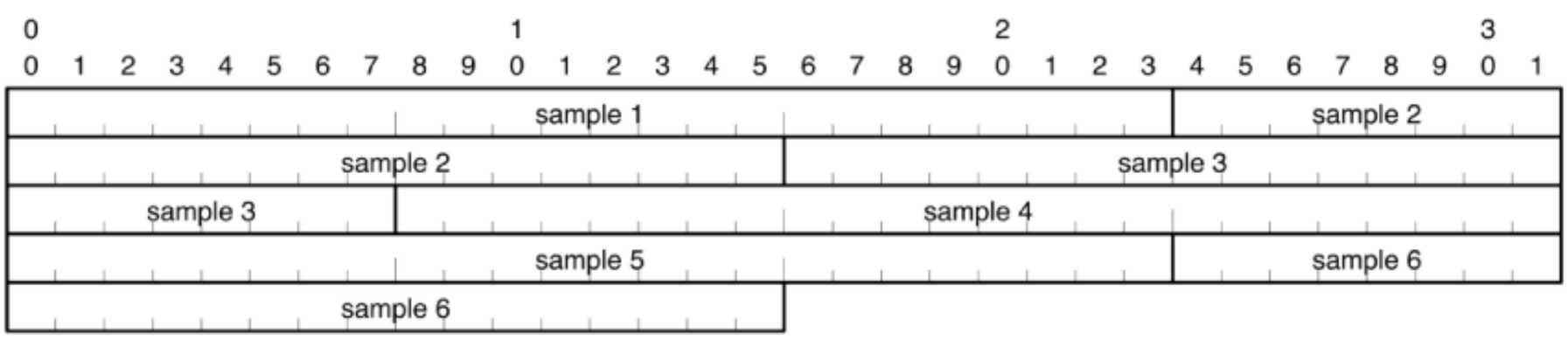


Figure 30—AAF 24-bit PCM integer, 1 channel, 6-samples-per-audio-sample frame

Copyright © 2016 IEEE. All rights reserved.

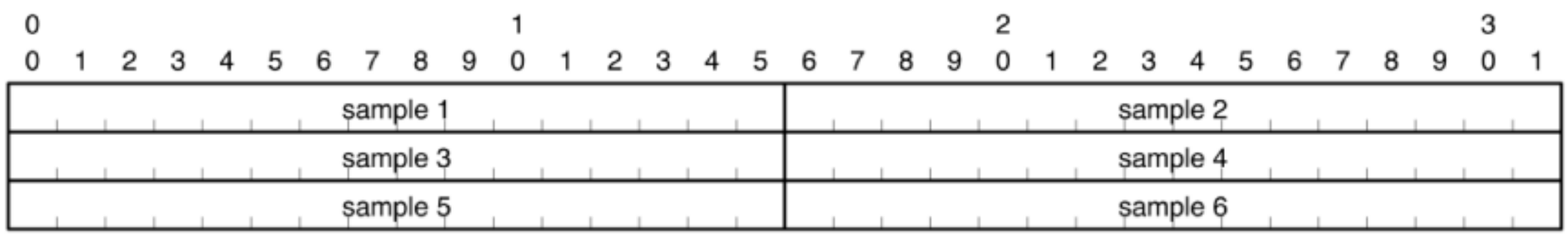


Figure 31—AAF 16-bit PCM integer, 1-channel, 6-samples-per-audio sample frame

7.3.6 AAF PCM audio channel assignments

The audio channel assignments for stereo, 5.1, and 7.1 PCM program types shall be mapped within an AAF stream according to Table 12, Table 13, and Table 14, respectively. These assignments are the same as used in SMPTE ST 2035:2009 [B30].

Other channel assignments for these (or other) stream channel counts can be used, however such channel assignments are outside the scope of this standard. For example, a stream with a **channels_per_frame** field value of 6 can contain audio that is not a 5.1 program type, but any such channel assignment information must be known a priori or communicated via another protocol (for example AVDECC).

Table 12—AAF stereo channel layout

Channel	Description
0	Left
1	Right

Table 13—AAF 5.1 channel layout

Channel	Description
0	Front left
1	Front right
2	Front center
3	Low frequency
4	Surround left
5	Surround right

Table 14—AAF 7.1 channel layout

Channel	Description
0	Front left
1	Front right
2	Front center
3	Low frequency
4	Surround left
5	Surround right
6	Surround back left
7	Surround back right

Copyright © 2016 IEEE. All rights reserved.

7.4 AAF AES3 stream data encapsulation

7.4.1.1 Overview

When the value of the **format** field is set to AES3_32BIT (see Table 9), the format of the AAF AVTPDU is given in this clause.

This encapsulation uses the **aaf_format_specific_data_1**, **afsd**, **aaf_format_specific_data_2**, and **aaf_format_specific_payload** fields of the AAF common header (Figure 25) to define the following specific fields:

— **nfr** (nominal frame rate): 4 bits

— **rsv** (reserved): 2 bits

— **streams_per_frame:** 10 bits

— **aes3_data_type_h:** 1 octet

— **aes3_dt_ref:** 3 bits

— **aes3_data_type_l:** 1 octet

— **aes3_data_payload:** 0 to *n* octets

These fields are shown in Figure 32.

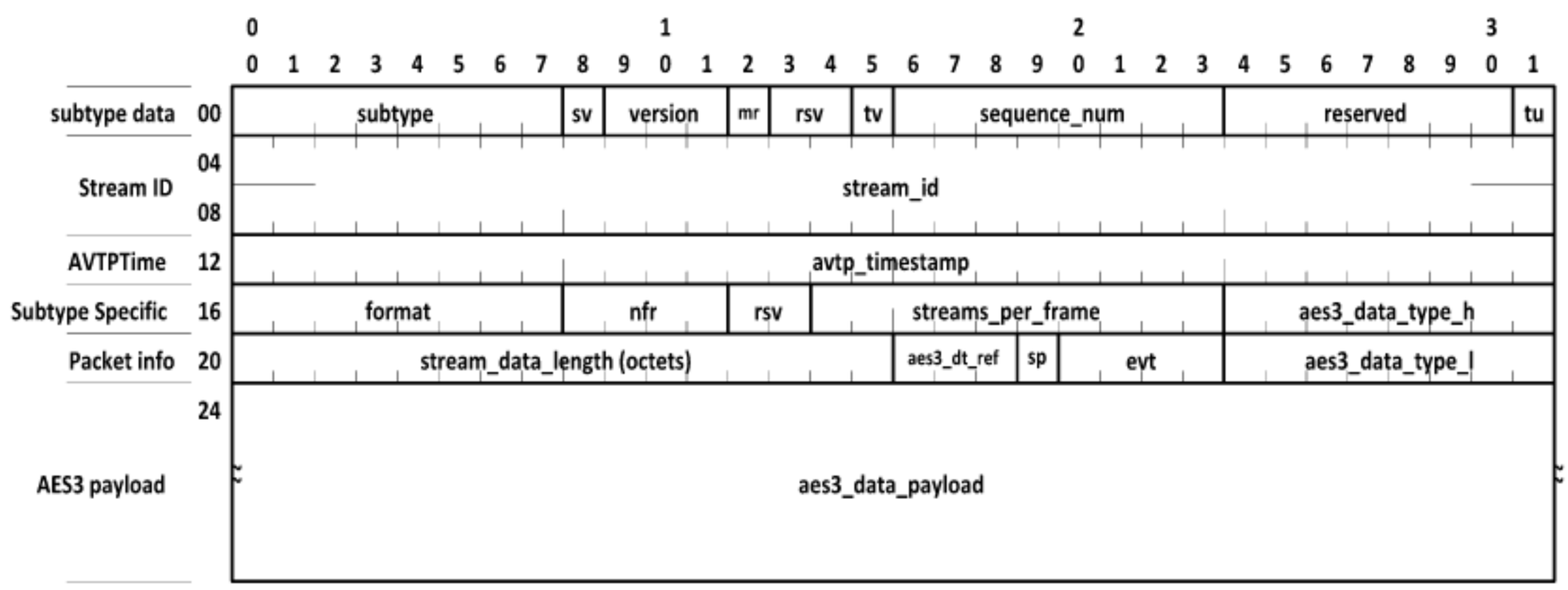


Figure 32 —AAF AES3 AVTPDU Format

7.4.2 nfr (nominal frame rate) field

The **nfr** field describes the AES3 frame rate (regardless of the sample rate of any audio samples carried within the AES3 frames). The **nfr** field shall be set to one of the values in Table 15.

Copyright © 2016 IEEE. All rights reserved.

Table 15—nfr field values

Value	Description
0_{16}	User specified
1_{16}	8 kHz
2_{16}	16 kHz
3_{16}	32 kHz
4_{16}	44.1 kHz
5_{16}	48 kHz
6_{16}	88.2 kHz
7_{16}	96 kHz
8_{16}	176.4 kHz
9_{16}	192 kHz
A_{16}	24 kHz
B to F_{16}	Reserved

7.4.3 streams_per_frame field

This field shall be set to the number of AES3 streams being carried by the AAF AVTPDU, a positive integer greater than zero (0). Only integral numbers of AES3 streams can be transmitted using AAF. For example, in the case of 3 AES3 streams being transmitted, the **streams_per_frame** field would be set to 3 (see Figure 33). For additional information on the required value for this field when using specific AES3 stream settings, see Annex K.

7.4.4 aes3_data_type field

7.4.4.1 General

The **aes3_data_type_l** and **aes3_data_type_h** fields (collectively the **aes3_data_type** field) can optionally be used to indicate the format of the audio contained within an AES3 stream. The **aes3_data_type_h** field contains the most-significant octet and the **aes3_data_type_l** field contains the least-significant octet of a single doublet value (see 3.4.1).

The **aes3_data_type** value is defined by one of various reference sources, for example the AES professional format or the S/PDIF consumer format (for further info see K.5). The value in the **aes3_dt_ref** field determines which reference standard is in use (see 7.4.5).

If more than one AES3 stream is contained in the AAF stream, then the value of the **aes3_data_type** field applies to all of the AES3 streams, with the exception of when the **aes3_dt_ref** field is set to DT_VENDOR.

7.4.4.2 aes3_data_type value ranges

The SMPTE ST 338 data type values are 5 bits wide. The IEC 61937 data type values are 7-bit wide values composed of a 5-bit data type (most-significant bits of **aes3_data_type** value) aggregated with a 2-bit subdata type (least-significant bits of **aes3_data_type** value). The **aes3_data_type** field is 2 octets wide to allow for future expansion of these tables.

Copyright © 2016 IEEE. All rights reserved.

7.4.5 aes3_dt_ref (aes3_data_type reference) field

The **aes3_dt_ref** field indicates which standard the **aes3_data_type** value (7.4.4) is based on. The **aes3_dt_ref** field shall be set to one of the values in Table 16.

Table 16—aes3_dt_ref field values

Value	Name	Description
0_{10}	DT_UNSPECIFIED	Data type not specified (**aes3_data_type** field is set to zero (0) by Talker and ignored by Listener)
1_{10}	DT_PCM	Data type is PCM (**aes3_data_type** field is set to zero (0) by Talker and ignored by Listener)
2_{10}	DT_SMPTE338	Data type reference is SMPTE ST 338 (professional application)
3_{10}	DT_IEC61937	Data type reference is IEC 61937-2 (consumer application [S/PDIF])
4_{10}	DT_VENDOR	Data type reference is defined by vendor
5 to 7_{10}	Reserved	___

When **aes3_dt_ref** is set to DT_UNSPECIFIED it indicates that an **aes3_data_type** field value is not provided, and the data type reference information must be known a priori or via another protocol such as AVDECC. In this case the **aes3_data_type** shall be set to zero (0) on transmit and ignored by the Listener.

When **aes3_dt_ref** is set to DT_PCM it indicates that the format of the AES3 stream is linear PCM. In this case the **aes3_data_type** field shall be set to zero (0) on transmit and ignored by the Listener.

When **aes3_dt_ref** is set to DT_SMPTE338 the data type reference is specified by SMPTE ST 338, Section 4, Table 1.

When **aes3_dt_ref** is set to DT_IEC61937 the data type reference is specified by IEC 61937-2:2011 Amend. 1, Section 4.2, Table 2.

Both data type tables include values that are undefined or otherwise do not specify a data type; such values shall not be used as values for the **aes3_data_type** field.

When **aes3_dt_ref** is set to DT_VENDOR it indicates that the vendor has defined their own set of **aes3_data_type** values.

NOTE—A DT_VENDOR value can be defined by the vendor to indicate a specific arrangement of a set of AES3 streams of differing data types within a single AAF stream, for example as defined by ISO 10681-1:2010 [B14], Section 5.2, Table 1. Alternatively, a DT_VENDOR data type can be defined by the vendor to indicate that the Listener is expected to parse the individual AES3 streams to determine their individual data types. Alternatively, unique data type (and other) information for each of multiple AES3 streams within a single AAF stream can be transmitted using AVDECC (see K.9).

7.4.6 aes3_data_payload field

7.4.6.1 Overview

The **aes3_data_payload** field contains the AAF subframes. All AAF AVTPDUs within a stream shall contain the same number of subframes. The byte order of the subrames shall be network byte order, i.e., most-significant byte at least AVTPDU byte offset.

NOTE—Because all AAF AES3 AVTPDUs in a stream are required to have the same number of subframes, the last AES3 AVTPDU in a stream cannot be shorter than previous AES3 AVTPDUs. That means that if there are not enough

Copyright © 2016 IEEE. All rights reserved.

subframes to fill an entire AAF AVTPDU, then the Talker can either drop (not transmit) the few remaining subframes or pad the remainder of the AAF AVTPDU with synthesized subframes. The values of such synthesized subframes are outside the scope of this standard.

If the **streams_per_frame** field is set to a value greater than one (1), then the subframes are interleaved chronologically into frames. For example, Figure 33 shows the layout of AES3 subframe quadlets in a three-streams-per-packet **aes3_data_payload** field.

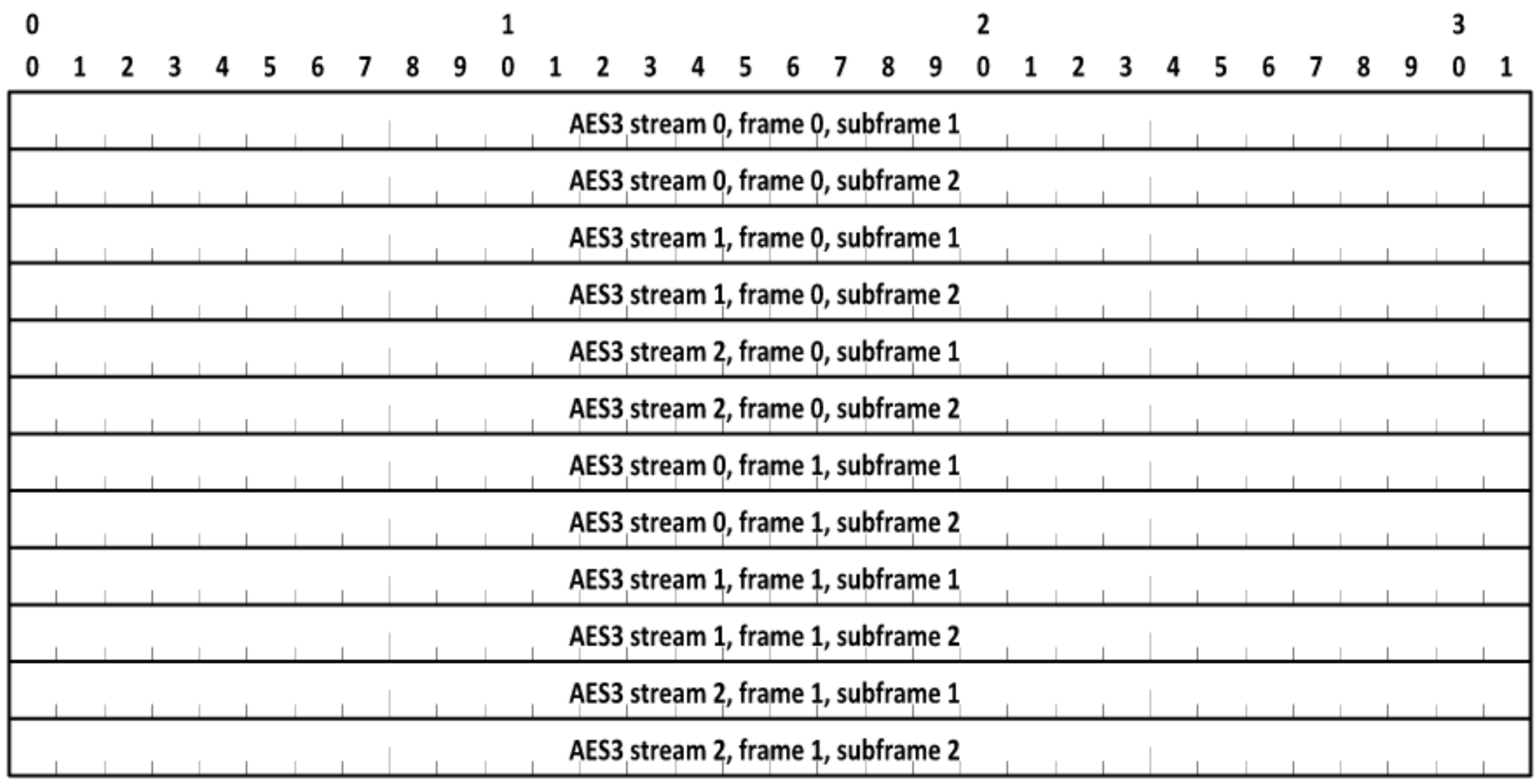


Figure 33—AAF AES3 payload, three AES3 streams

7.4.6.2 AAF subframe

The AES3 payload of an AAF stream consists of AAF subframes that are similar to (but not identical to) subframes as defined by AES3, as shown in Figure 34. (For reference the AES3 subframe format is shown in Figure K.2.)

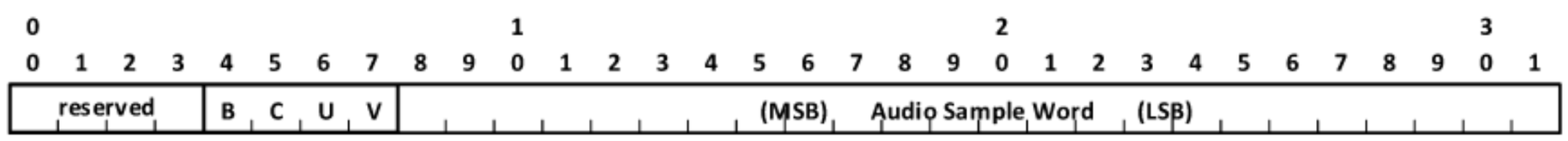


Figure 34—AAF Subframe Format

NOTE—AES3 subframes to be transmitted via AAF must be reformatted as described in this subclause to match the AAF subframe format.

The bit order of the audio sample word follows the AVTP convention (most-significant bit at least offset, see Figure 1) and the audio sample word and ancillary data bits are aligned as shown in Figure 34.

The AES3 serial preambles are not transmitted, and bits 0 to 3 (marked "reserved" in Figure 34, and marked "Preamble" in Figure K.2) shall be set to zero (0) on transmit and ignored by the Listener.

Copyright © 2016 IEEE. All rights reserved.

The sequence of the subframes in the payload shall be subframe 1 followed by subframe 2 for each AES3 frame (see Figure 33).

The audio sample data words in the subframes follow the AES3 convention of not specifying specific treatment of insignificant bits. That is, they are not intended to be set to 0 before transmission, as they would be with PCM samples in AAF (as described in 7.3.4). All bits of the AES3 24-bit audio sample word field shall be transmitted without modification (see K.10.7).

7.4.6.3 B (block start) bit

The **B** bit shall be set to one (1) for the first subframe of the block specified in AES3-3, and to zero (0) for all other subframes (as defined in AES47-2006-r2011 Section 4.1.3.3). For additional information see Annex K.2.

7.4.6.4 C (channel status) bit

Each AES3 block conveys two 192-bit (24-octet) channel status data structures (CS), which shall be transmitted as a single data bit per subframe via the **C** bit (AES3-2-2009, Section 5.5). See also Annex K.2.

7.4.6.5 U (user data) bit

The **U** bit allows users to transmit another two (optional, user defined) 192-bit data structures per AES3 block (AES3-2-2009, Section 4).

7.4.6.6 V (validity) bit

The **V** bit shall be set to zero (0) to indicate that the format of the AES3 audio data is PCM audio, and set to one (1) to indicate that it is in some other format, for example coded audio such as Dolby E. See K.5.

7.5 AAF timestamps

The AAF common encapsulation makes use of the **avtp_timestamp** field to communicate an AVTP Presentation Time as described in 4.4.4.9.

If the **sp** field is set to zero (0), all AAF AVTPDUs shall set the **tv** (avtp_timestamp valid) field to one (1) and contain valid data in the **avtp_timestamp** field.

If the **sp** field is set to one (1), every eighth AAF AVTPDU shall set the **tv** (avtp_timestamp valid) field to one (1) and contain valid data in the **avtp_timestamp** field. All other AAF AVTPDUs shall set the **tv** (avtp_timestamp valid) field to zero (0) and the data in the **avtp_timestamp** field shall be ignored.

The **avtp_timestamp** field shall contain the AVTP Presentation Time of the first audio sample contained in the AAF AVTPDU as described in 4.3.2. If the AAF AVTPDU contains more than one (1) audio channel, the same AVTP Presentation Time shall apply to all channels of the first audio sample frame in the AAF AVTPDU.

Copyright © 2016 IEEE. All rights reserved.

7.6 AAF packetization

AAF uses a packetization scheme that is different from IEC 61883. The transmit interval of AVTPDUs within an AAF stream is determined by the Talker, according to the rate of the media sample clock, in order to maintain a consistent number of audio sample frames per AAF AVTPDU (see 7.3.5).

As shown in Figure 35, the AVTP audio packetization results in the same number of audio sample frames in every AAF AVTPDU of a stream. The transmission interval of the AAF AVTPDUs within the stream is driven by the number of channels, the payload data size, and the media clock rate (see 4.3).

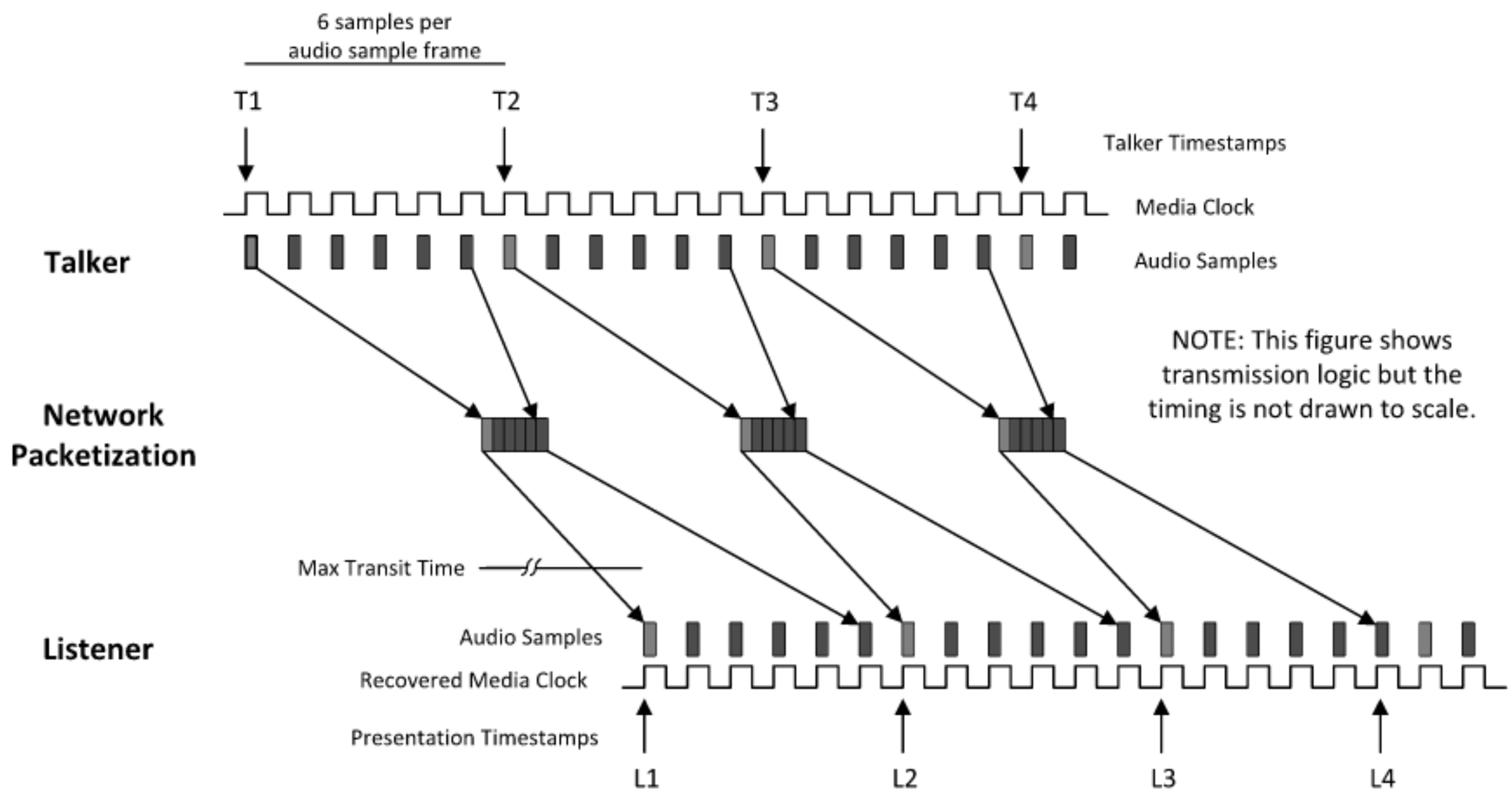


Figure 35—AVTP audio packetization example

7.7 AAF and SRP

AAF requires that the same number of audio sample frames are present in every AAF AVTPDU. Therefore the AAF transmission interval is defined by the clock rate of the media rather than the FQTSS observation interval.

The SRP reservation is performed based on the FQTSS observation interval and the AVTP Audio transmission interval. The reservation is the maximum number of AAF AVTPDUs transmitted in an FQTSS observation interval when both are at nominal rates.

For example, a 48-kHz stream with 6 samples per AAF AVTPDU and an FQTSS observation interval of 125 μs, the SRP reservation will be for 1 frame per observation interval since the transmission interval is also 125 μs. A 44.1-kHz stream with 6 samples per AAF AVTPDU and an FQTSS observation interval of 125 μs also has an SRP reservation of 1 frame per observation interval even though there will periodically be an observation interval where no AAF AVTPDU will be transmitted since it has a transmission interval of 136.054 μs as can be seen in the example given in Figure 36.

Copyright © 2016 IEEE. All rights reserved.

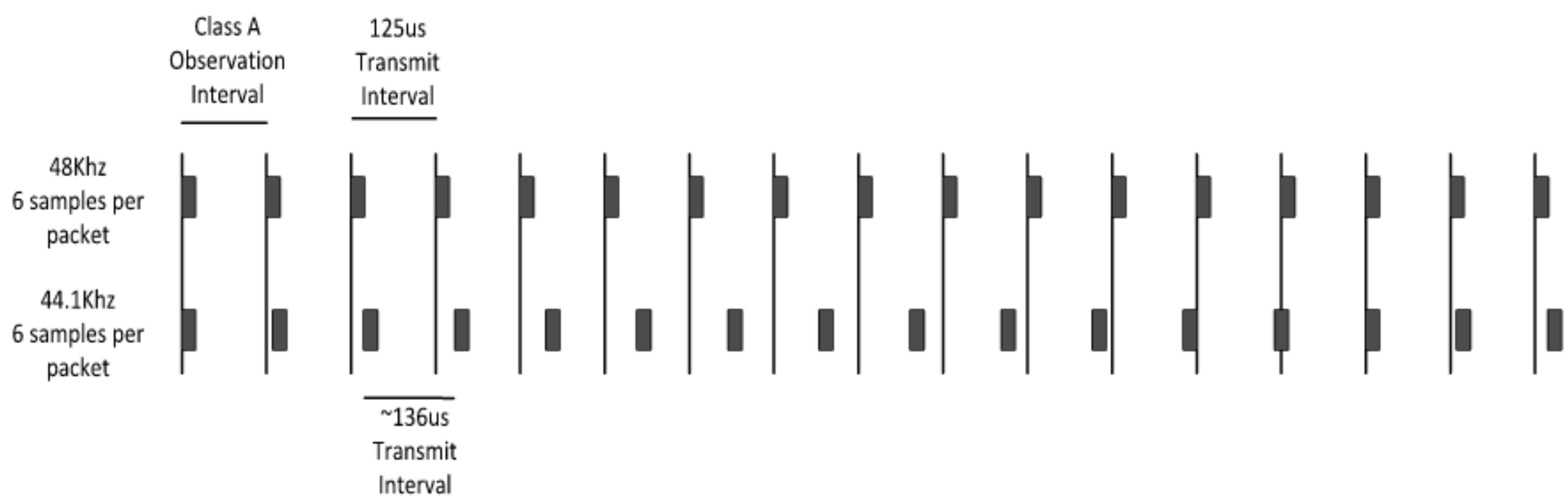


Figure 36—AVTP audio SRP example

In order to maintain interoperability between devices, the transmission intervals listed in Table 17 and Table 18 should be used. The transmit intervals given are in the media clock time domain, not the gPTP time domain; the exact transmit interval can be calculated from the media clock rate and the audio sample frames per AAF AVTPDU.

Table 17—Recommended transmit interval for 125 µs FQTSS observation interval

Media clock rate	Audio sample frames/AAF AVTPDU	Approximate transmit interval
8 kHz	1	125 µs
16 kHz	2	125 µs
32 kHz	4	125 µs
44.1 kHz	6	136 µs
48 kHz	6	125 µs
88.2 kHz	12	136 µs
96 kHz	12	125 µs
176.4 kHz	24	136 µs
192 kHz	24	125 µs

Table 18—Recommended transmit interval for 250 µs FQTSS observation interval

Media clock rate	Audio sample frames/AAF AVTPDU	Approximate transmit interval
8 kHz	2	250 µs
16 kHz	4	250 µs
32 kHz	8	250 µs
44.1 kHz	12	272 µs
48 kHz	12	250 µs
88.2 kHz	24	272 µs
96 kHz	24	250 µs
176.4 kHz	48	272 µs
192 kHz	48	250 µs

NOTE—The default 125 µs and 250 µs FQTSS observation intervals are not the only possible observation intervals. Other transmit intervals are possible, but are outside of the scope of this standard.

Copyright © 2016 IEEE. All rights reserved.

8. Compressed Video Format

8.1 Overview

This clause describes the AVTPDU encapsulations for the Compressed Video Format (CVF).

CVF makes use of video payload definitions that are defined in the Internet Engineering Task Force IETF RFCs (see Table 20). To the extent possible, the CVF maintains the video data formatting from the original RFC. There are two significant differences from the RFC when these payloads are used in an AVTP network.

a) The formats make use of a Compressed Video Format header described in 8.3.

b) Timing and synchronization are based on AVTP timestamps with the AVTP timestamps representing AVTP Presentation Times as described in 4.4.4.9 and in the individual descriptions of these formats.

NOTE—Because video streams often have a variable bit rate, care must be taken when making reservations for them. Reservations for video streams must be sized for the maximum number of packets per class measurement interval. In addition to prevent impacting other streams on the network, all video Talkers must meet the credit-based shaping requirements for each stream's reservation as described in Clause 34 of IEEE Std 802.1Q-2014.

8.2 AVTP Presentation Time handling

8.2.1.1 Overview

In compressed video formats the packetized video units of the compressed video stream often exceed the maximum MTU size of an AVTPDU. When this occurs, the CVF Talker must fragment the video packetization unit for transmission on the AVB stream. Figure 37 shows the generic form of this packetization process for the Compressed Video Format.

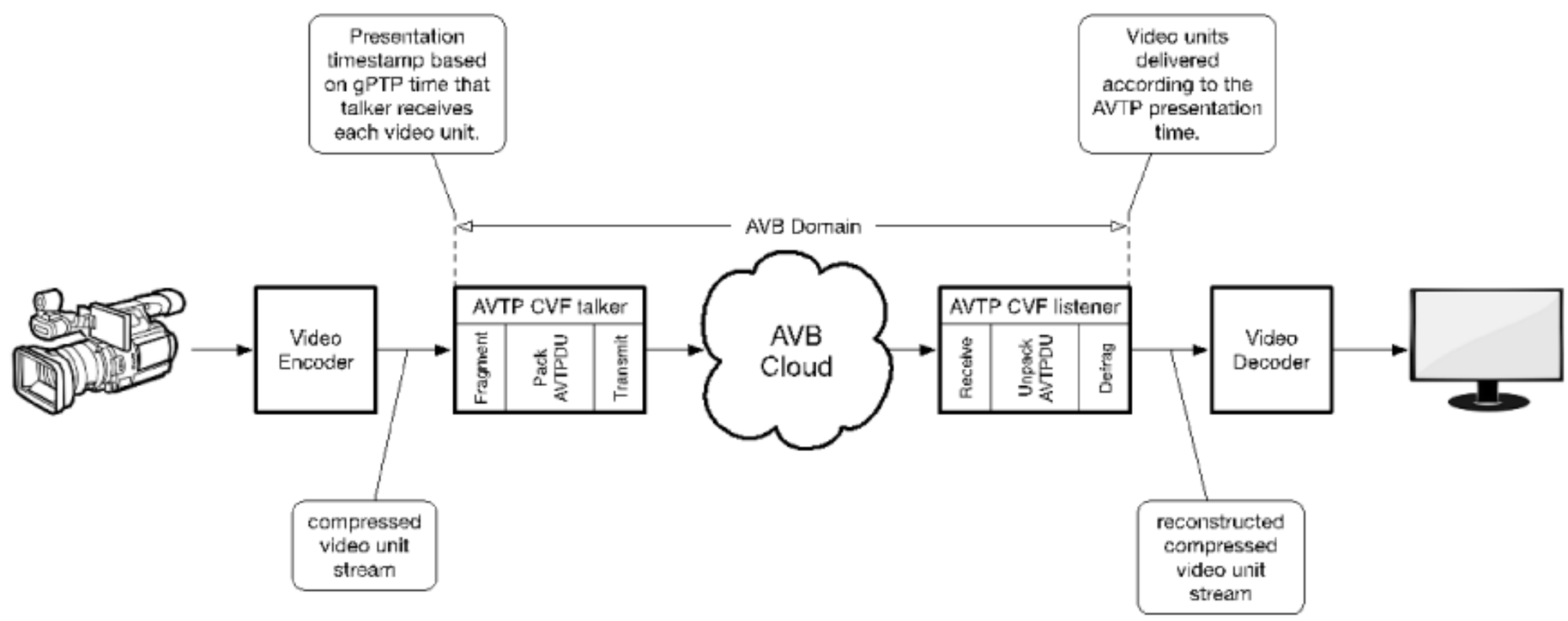


Figure 37—Generic packetization process for the Compressed Video Format

Copyright © 2016 IEEE. All rights reserved.

The technique used for fragmentation of the video packetization units is dependent on the specific video compression technique being used. Details on fragmentation can be found in subsequent clauses for each video compression technique.

When a CVF Talker fragments a video unit, it calculates the **avtp_timestamp** using the standard technique (see 4.3) based on the time it receives the complete video unit from the video encoding process. However, the CVF Talker only inserts this AVTP Presentation Timestamp in the **avtp_timestamp** field of the CVF AVTPDU carrying the *final fragment* of the video packetization unit. For all earlier fragments, the Talker marks the CVF AVTPDU's with an invalid timestamp (**tv** = 0). A CVF Listener assembles and presents each video packetization unit according to the AVTP Presentation Timestamp found in the CVF AVTPDU carrying the final video unit fragment. Using an invalid AVTP timestamp on earlier video unit fragments allows the Talker to transmit these CVF AVTPDUs without having to wait for the calculation of the final AVTP Presentation Time.

8.2.2 Video media clocking

The use of the **avtp_timestamp** field in the CVF AVTPDU guarantees that compressed video data is delivered to every Listener's video decompression and presentation engine at the same time. However it does not guarantee that every Listener presents the video (on the screen) at the same time. Each Listener's video decompression and presentation engine is unique and has its own internal delays. In addition, each Listener's video decoder re-creates the media clock used as the basis for the video presentation (typically a 90-kHz clock) independent from the Talker and from all other Listeners. Synchronous presentation of video is beyond the scope of this standard. IEEE Std 1722.1-2013 provides an example mechanism for managing and compensating for device-specific video decode and presentation differences.

It is recommended that another technique—such as the Clock Reference Format (CRF) described in Clause 10—be used to deliver and enable a common video frame clock. This technique is useful in applications where there are multiple video sources that must be synchronized, such as with an automotive multi-camera around-view system.

8.3 Compressed Video Stream data encapsulation

8.3.1 Overview

The Compressed Video Stream data encapsulation is used for carrying video streams defined by this subclause.

This encapsulation shall use the common stream header and shall have a **subtype** field of CVF (see 4.4.3.2).

This encapsulation uses the following common stream fields (see 4.4.3 and 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

Copyright © 2016 IEEE. All rights reserved.

— **stream_id**: 8 octets

— **avtp_timestamp**: 4 octets

— **stream_data_length**: 2 octets

This encapsulation defines the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, **format_specific_data_3**, and **stream_data_payload** fields of the common stream header to define the following format-specific fields:

— **rsv** (reserved): 2 bits

— **reserved**: 7 bits

— **format**: 1 octet

— **format_subtype**: 1 octet

— **reserved**: 2 octets

— **rsv** (reserved): 3 bits

— **M**: 1 bit

— **evt**: 4 bits

— **reserved**: 1 octet

— **video_data_payload**: 0 to *n* octets

These fields are shown in Figure 38.

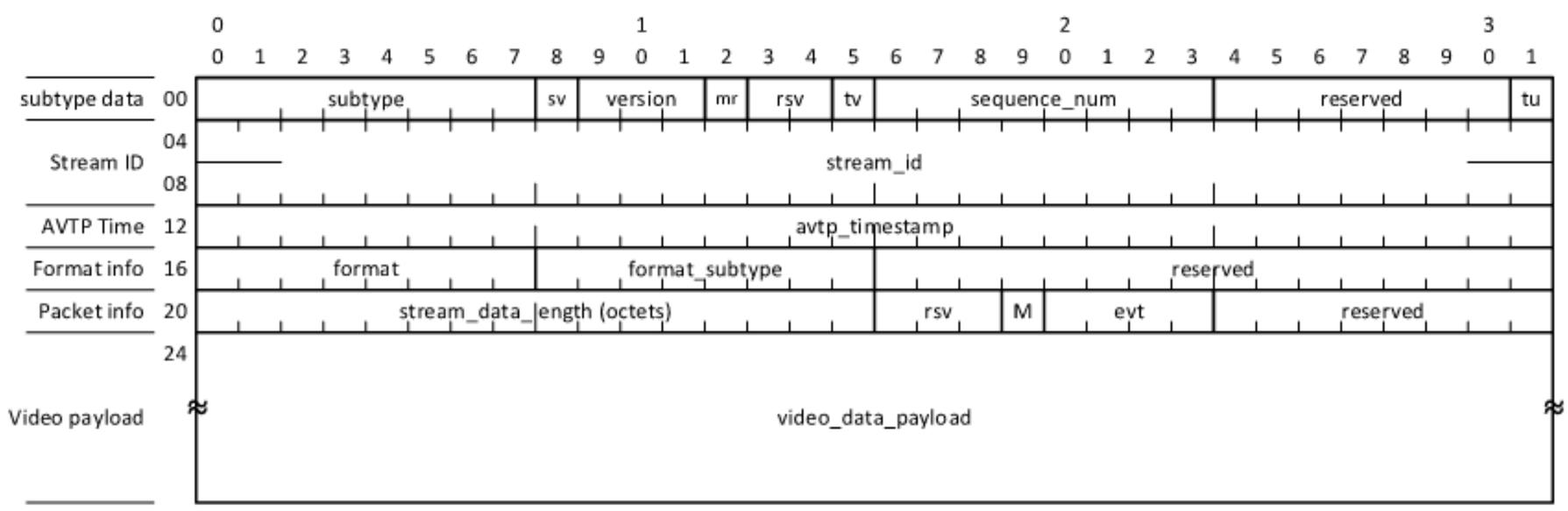


Figure 38—Compressed Video Format PDU Format

8.3.2 format field

The **format** field identifies the payload format of the CVF AVTPDU as defined in Table 19. If a CVF AVTPDU is received with a reserved **format**, it shall be ignored.

Copyright © 2016 IEEE. All rights reserved.

Table 19 —Compressed Video Format field

Value	Name	Description
0 to 1_{16}	Reserved	Reserved
2_{16}	RFC	RFC payload type
3 to FF_{16}	Reserved	Reserved

8.3.3 format_subtype field

The **format_subtype** field carries 1 octet of format subtype data as specified by the **format** field. The usage of the **format_subtype** field is defined by each CVF **format**.

When the **format** field is set to RFC, the **format_subtype** field is defined as shown in Table 20.

Table 20 —format_subtype field for RFC Format

Value	Name	Description	Clause
0_{16}	MJPEG	MJPEG Format (RFC 2435)	8.4
1_{16}	H264	H.264 Format (RFC 6184)	8.5
2_{16}	JPEG2000	JPEG 2000 Video (RFC 5371)	8.6
3 to FF_{16}	Reserved	Reserved	—

8.3.4 M field

The **M** field is defined on a **format**/**format_subtype** specific basis. If a **format** does not define a usage for **M**, then the **M** field shall be set to zero (0).

8.3.5 evt field

The **evt** field contains 4 bits that are reserved for upper-level protocols to indicate events in the stream. The specific use of these bits is outside the scope of this standard.

8.4 MJPEG Video Format

8.4.1 Overview

The MJPEG Video Format uses a **format** value of RFC (see Table 19) and a **format_subtype** value of MJPEG (see Table 20).

This format uses the **video_data_payload** field to define the following fields:

— **type_specific**: 1 octet

— **fragment_offset**: 3 octets

— **type**: 1 octet

— **Q**: 1 octet

— **width**: 1 octet

Copyright © 2016 IEEE. All rights reserved.

— **height**: 1 octet

— **JPEG_payload**: 0 to *n* octets

The fields of the MJPEG Video Format are given in Figure 39.

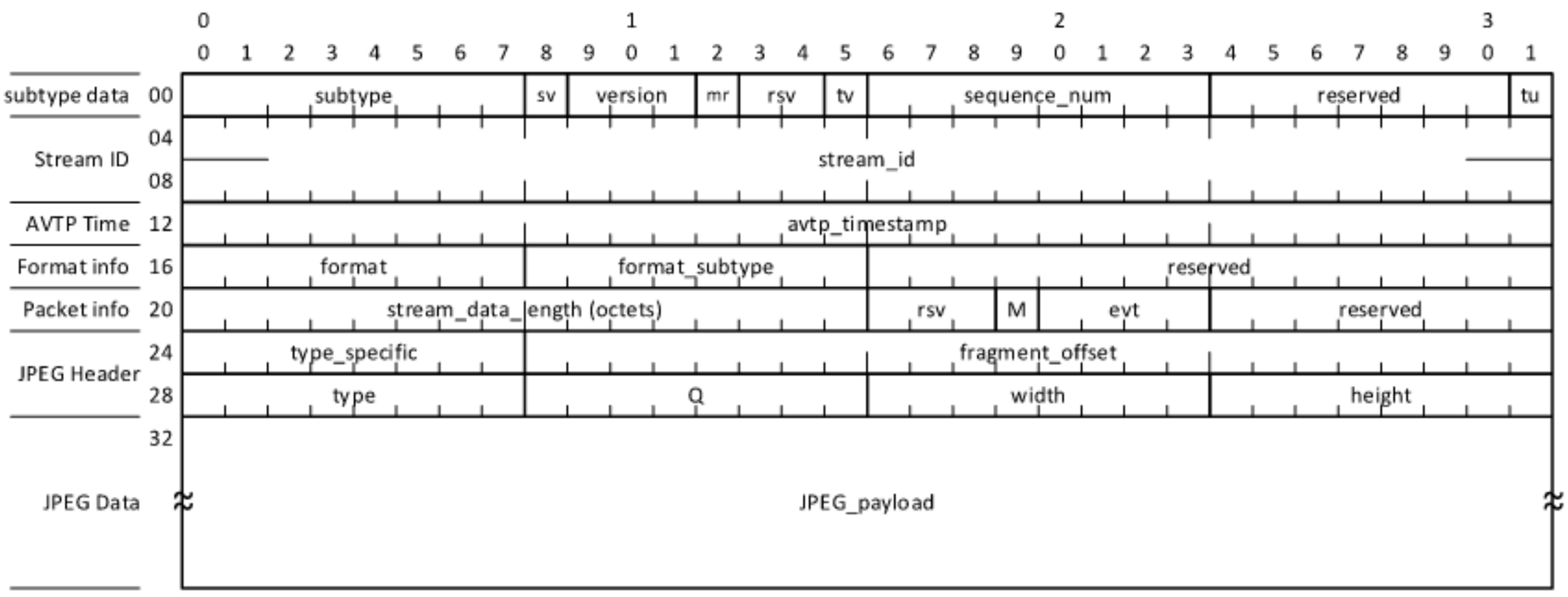


Figure 39 —The MJPEG Compressed Video Format PDU

8.4.2 Common header data encapsulation

8.4.2.1 Overview

The MJPEG Video Format uses the defined Compressed Video Format header fields as defined in Figure 38, with the specific usage defined in this subclause.

8.4.2.2 avtp_timestamp field

The **avtp_timestamp** represents the AVTP Presentation Time associated with the given video packetization unit. It is based on the gPTP time at which the MJPEG Talker receives each MJPEG video unit and is calculated according to the techniques described in 4.4.4.9. The Talker shall fill the **avtp_timestamp** with this AVTP Presentation Time, but only in the MJPEG AVTPDU carrying the final fragment of the video packetization unit. MJPEG AVTPDUs carrying pre-final video packetization unit fragments shall be marked with an invalid **avtp_timestamp** by setting the timestamp_valid field to zero (**tv** = 0).

8.4.2.3 M field

The **M** marker bit shall be set in the last packet of a video frame.

Copyright © 2016 IEEE. All rights reserved.

8.4.3 Additional header and data fields

8.4.3.1 Overview

The MJPEG Video Format uses the following additional header and data fields.

8.4.3.2 type_specific field

See RFC 2435 Section 3.1.1.

8.4.3.3 fragment_offset field

See RFC 2435 Section 3.1.2.

8.4.3.4 type field

See RFC 2435 Sections 3.1.3 and 4.1.

8.4.3.5 Q field

See RFC 2435 Section 3.1.4 and 4.2.

8.4.3.6 width field

See RFC 2435 Section 3.1.5.

8.4.3.7 height field

See RFC 2435 Section 3.1.6.

8.4.3.8 JPEG_payload field

See RFC 2435 Sections 3.1.7, 3.1.8, 3.1.9, and 4.

8.4.4 Packetization

MJPEG video frames can be larger than a single CVF AVTPDU, and as such, MJPEG video frames may be fragmented into multiple MJPEG AVTPDUs. If support for partial frame decoding is desired, frames may be fragmented such that each packet contains an integral number of restart intervals (see RFC 2435 Section 4.4).

Each MJPEG AVTPDU that makes up a single JPEG video frame shall have its **avtp_timestamp** and **tv** fields set according to 8.4.2.2. The fragment offset field of each MJPEG AVTPDU shall be set to the byte offset of its payload data within the original video frame. The **M** bit shall be set on the last MJPEG AVTPDU of a video frame. MJPEG AVTPDUs of a video frame are sent sequentially and the fragments they contain shall not overlap.

Copyright © 2016 IEEE. All rights reserved.

A given MJPEG AVTPDU shall contain video frame data from a single JPEG video frame.

For more information, see RFC 2435 Section 4.

8.5 H.264 Compressed Video Format

8.5.1 Overview

The H.264 Compressed Video Format uses a **format** value of RFC (see Table 19) and a **format_subtype** value of H264 (see Table 20).

This format uses the **video_data_payload** field to define the following fields:

— **h264_timestamp**: 4 octets

— **h264_payload**: 0 to *n* octets

The fields of the H.264 Compressed Video Format are given in Figure 40.

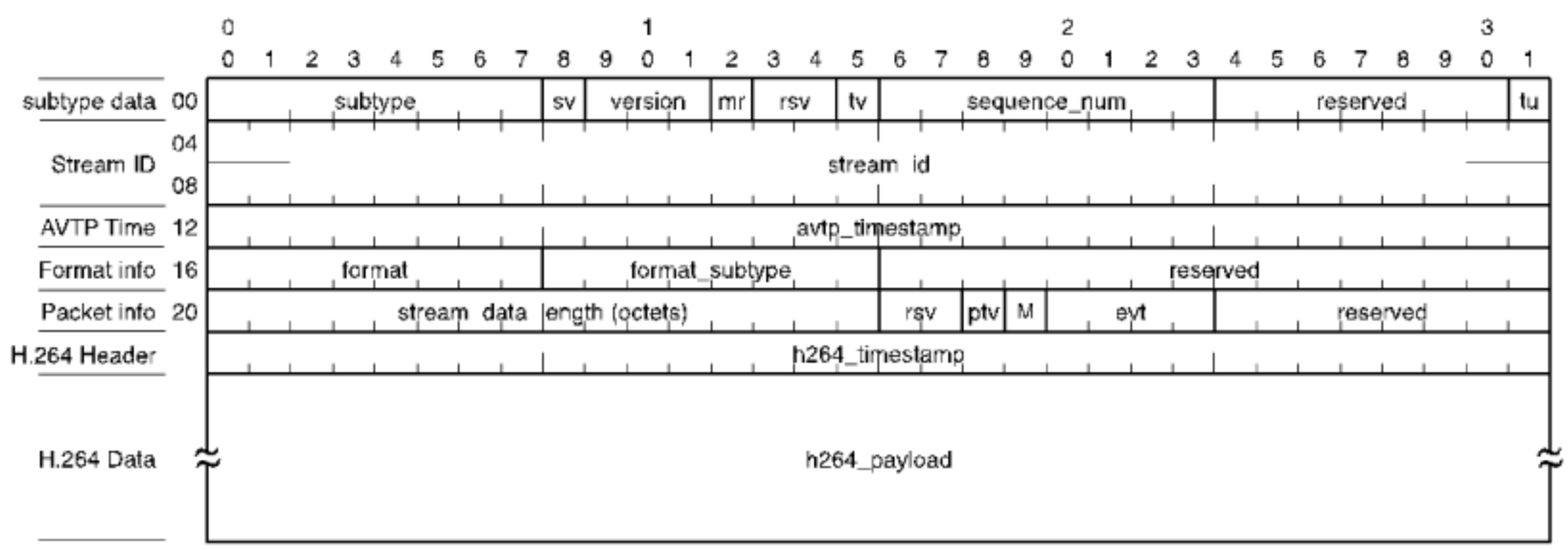


Figure 40—H.264 Compressed Video Format PDU

8.5.2 Common header data encapsulation

8.5.2.1 Overview

The H.264 video format uses the defined Compressed Video Format header fields as defined in Figure 38, with specific usage defined in this subclause.

8.5.2.2 avtp_timestamp field

The **avtp_timestamp** represents the AVTP Presentation Time associated with the given H.264 Network Abstraction Layer (NAL) unit. It is based on the gPTP time at which the H.264 Talker receives the H.264 NAL unit and is calculated according to the techniques described in 4.4.4.9. The Talker shall fill the **avtp_timestamp** with this AVTP Presentation Time, but only in the H.264 AVTPDU carrying the final fragment of the video packetization unit. H.264 AVTPDUs carrying pre-final video packetization unit fragments shall be marked with an invalid **avtp_timestamp** by setting the timestamp_valid field to zero (**tv** = 0).

Copyright © 2016 IEEE. All rights reserved.

NOTE—The **avtp_timestamp** field does not carry the timing information produced during the H.264 encoding process and that is associated with decoding and rendering/presentation of each H.264 NAL unit. That timestamp is carried in the **h264_timestamp** field (see 8.5.3.1).

8.5.2.3 ptv field

The **ptv** field indicates the validity of the payload timestamp field. In the case of the H.264 AVTPDU, it indicates the validity of the **h264_timestamp** field.

The H.264 CVF Talker shall set the **ptv** bit to one (1) if the **h264_timestamp** field contains a valid value, or zero (0) otherwise.

The H.264 CVF Listener shall ignore the value of the **h264_timestamp** field if **ptv** is set to zero (0).

NOTE—H.264 CVF Talkers are encouraged to insert valid H.264 timestamps to provide valuable assistance to the H.264 decoder, eliminate the need for the decoder to recreate NAL unit timestamps, and thereby reduce decoding complexity and latency. Often NAL unit timestamp information is available from the H.264 encoding process; H.264 CVF Talkers should make every effort to transmit this information in the H.264 AVTPDU.

8.5.2.4 M field

The **M** field shall be set according to RFC6184, Section 5.1.

8.5.3 Additional header and data fields

8.5.3.1 h264_timestamp field

The **h264_timestamp** field carries the timestamp information associated with the H.264 NAL unit fragment. It is used by the H.264 decode process to properly synchronize, decode, and present H.264 video frames. The **h264_timestamp** shall be calculated according to RFC6184, Section 5.1.

8.5.4 H.264 payload

The **h264_payload** field shall be constructed according to RFC6184 Sections 5.2 through5.8, 6, and 7. This includes payloads that are constructed of a single H.264 NAL unit (RFC6184, Section 5.6), of an aggregation of H.264 NAL units (RFC6184, Section 5.7), and of a fragmentation of a large H.264 NAL unit (RFC6184, Section 5.8). Examples of these AVTPDUs are shown in Figure 41, Figure 42, Figure 43, Figure 44, Figure 45, and Figure 46.

In each of these diagrams, the AVTP Video Header block represents all of the header fields shown in Figure 40, from the **subtype** field to the **reserved** field at octet offset 23, inclusively. The **h264_timestamp** field is explicitly shown in each figure.

Copyright © 2016 IEEE. All rights reserved.

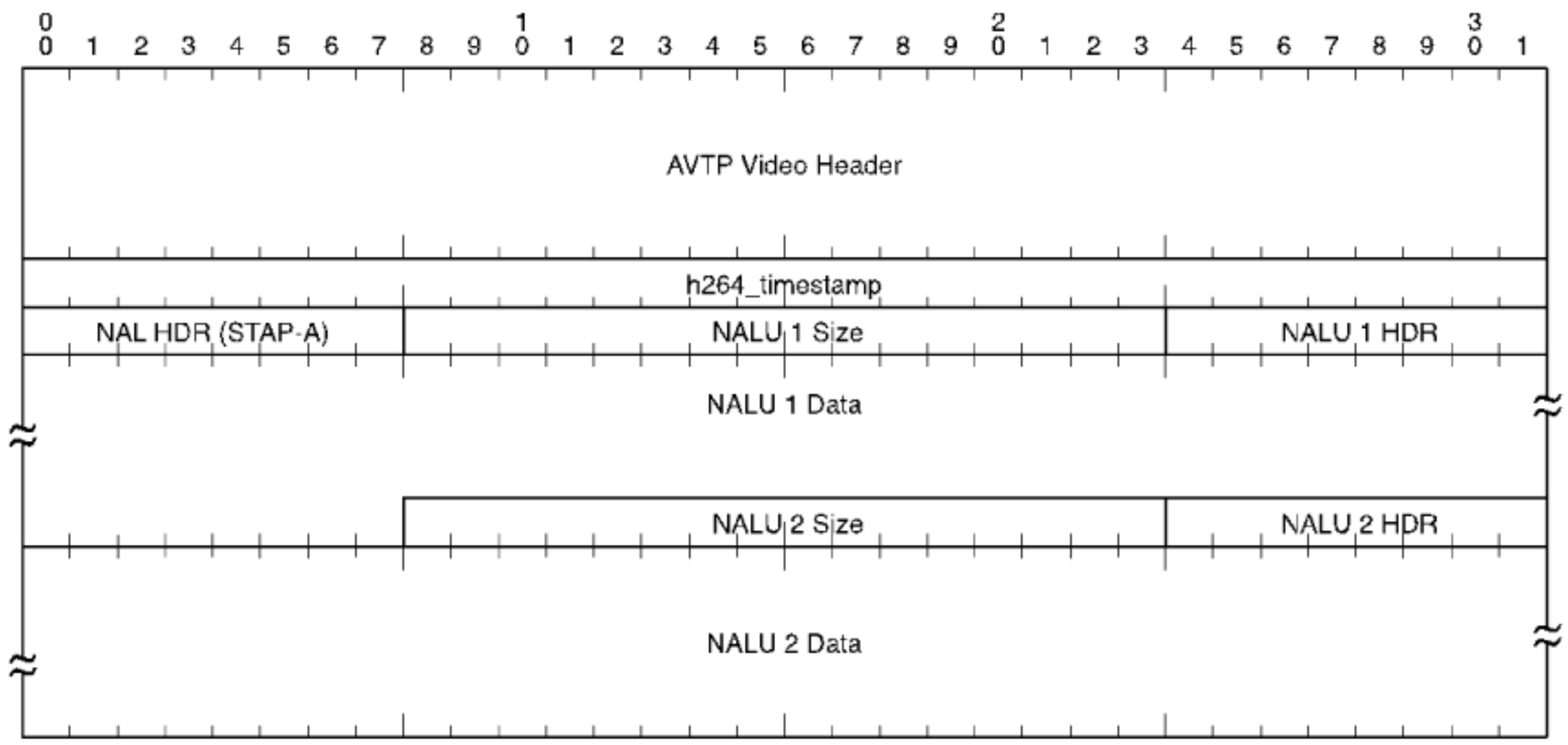


Figure 41—Example of an AVTP packet including an STAP-A containing two single-time aggregation units

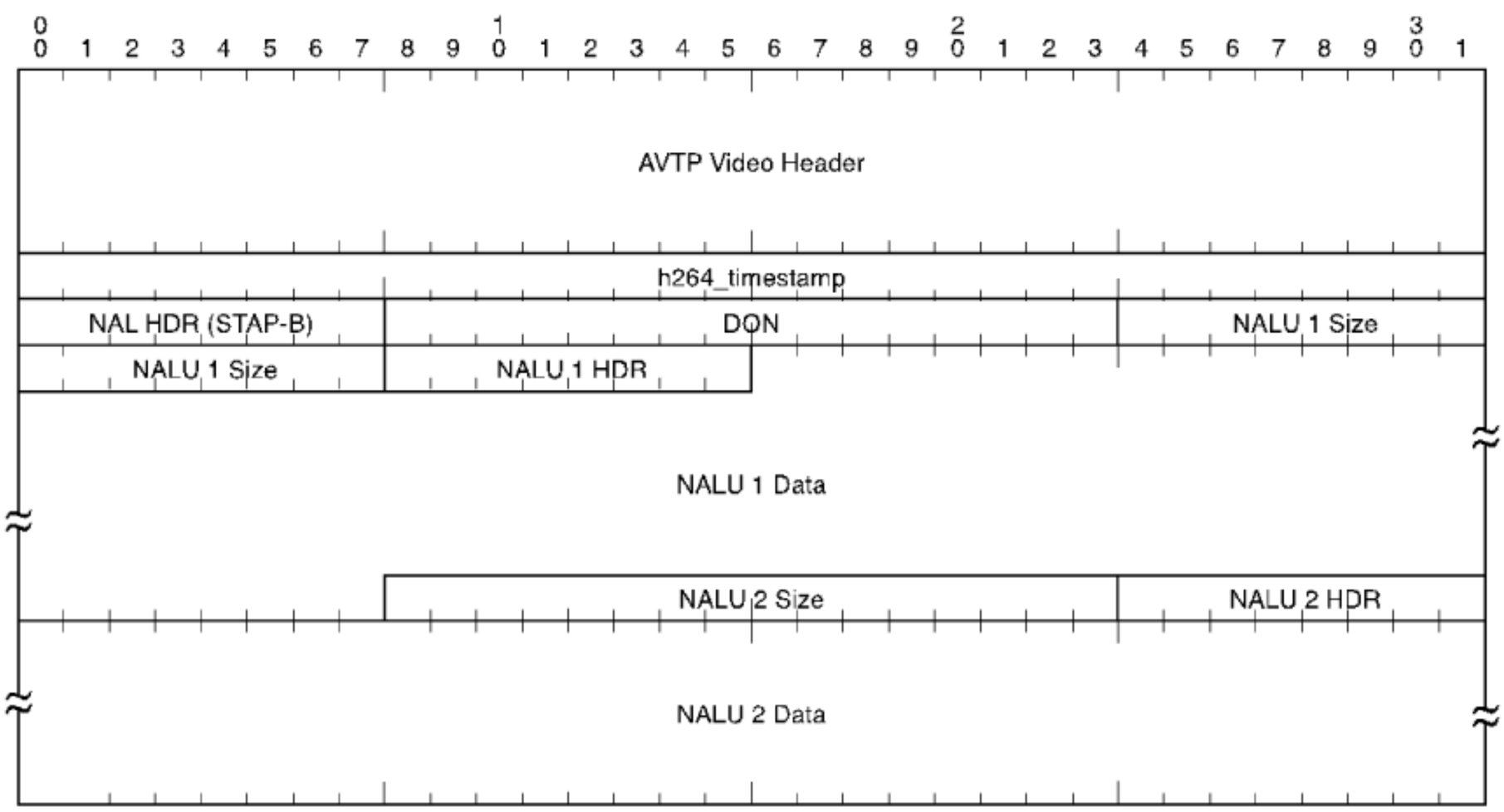


Figure 42—Example of an AVTP packet including an STAP-B containing two single-time aggregation units

Copyright © 2016 IEEE. All rights reserved.

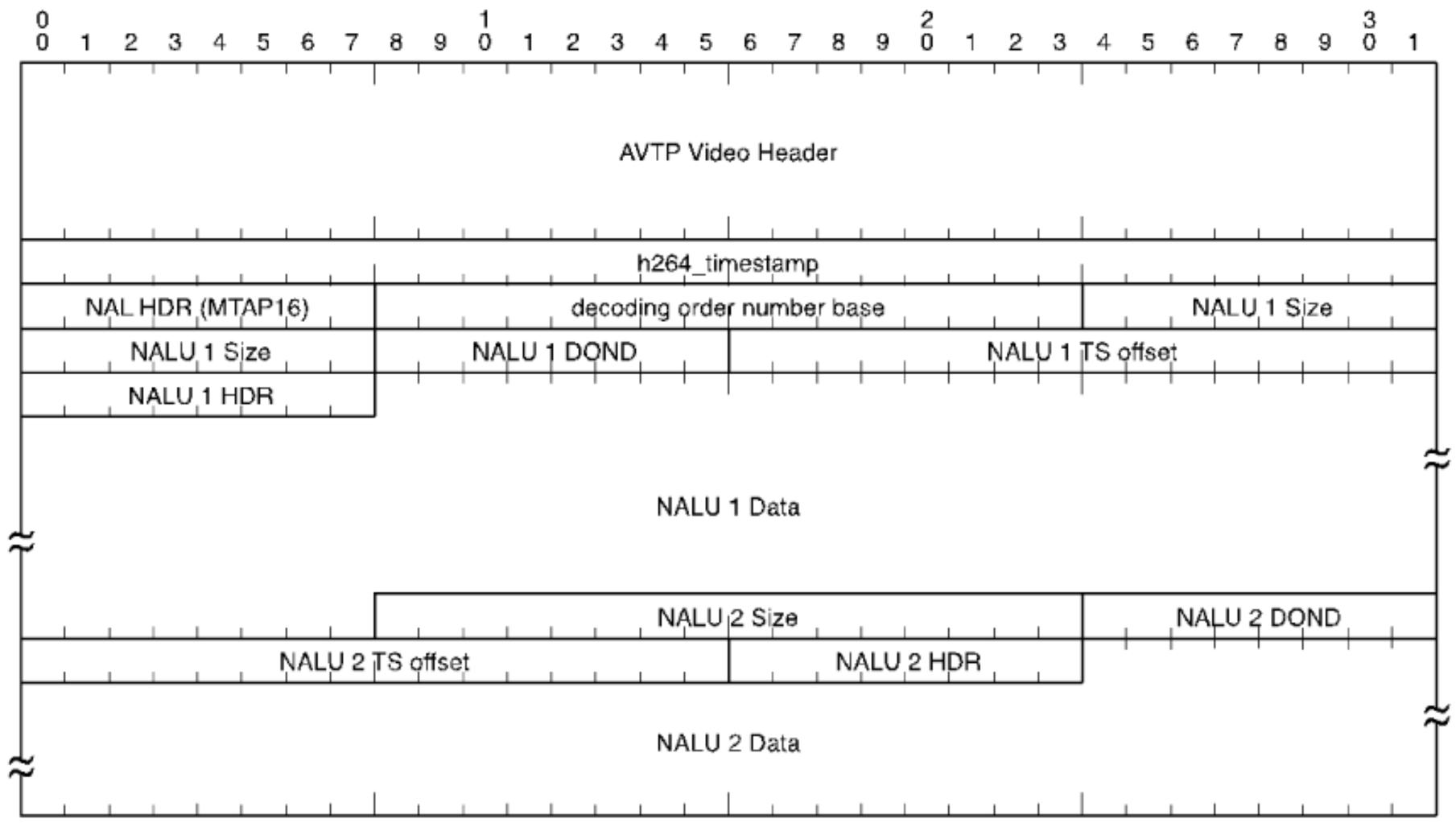


Figure 43—AVTP packet including a multi-time aggregation packet of type MTAP16 containing two multi-time aggregation units

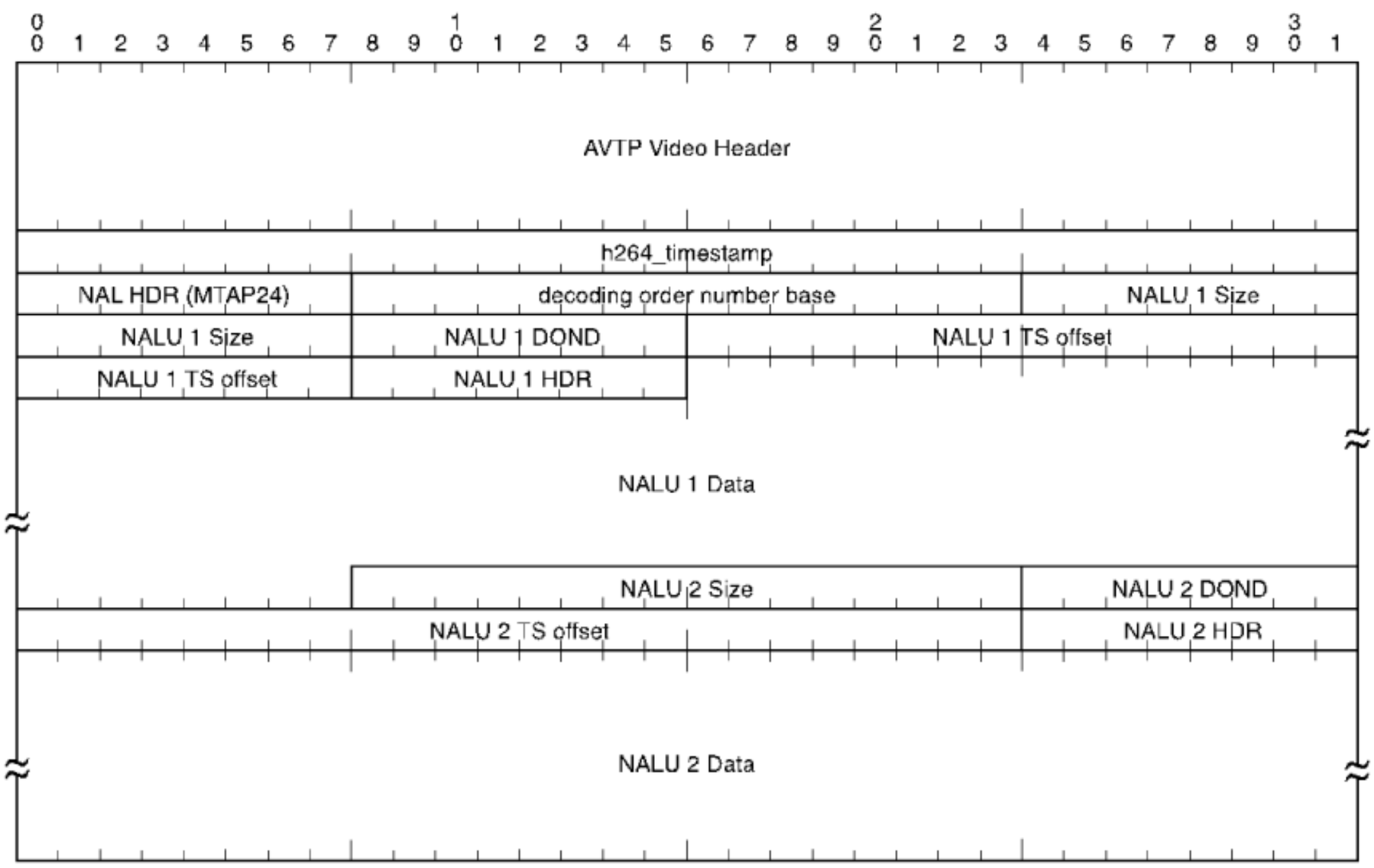


Figure 44—AVTP packet including a multi-time aggregation packet of type MTAP24 containing two multi-time aggregation units

Copyright © 2016 IEEE. All rights reserved.

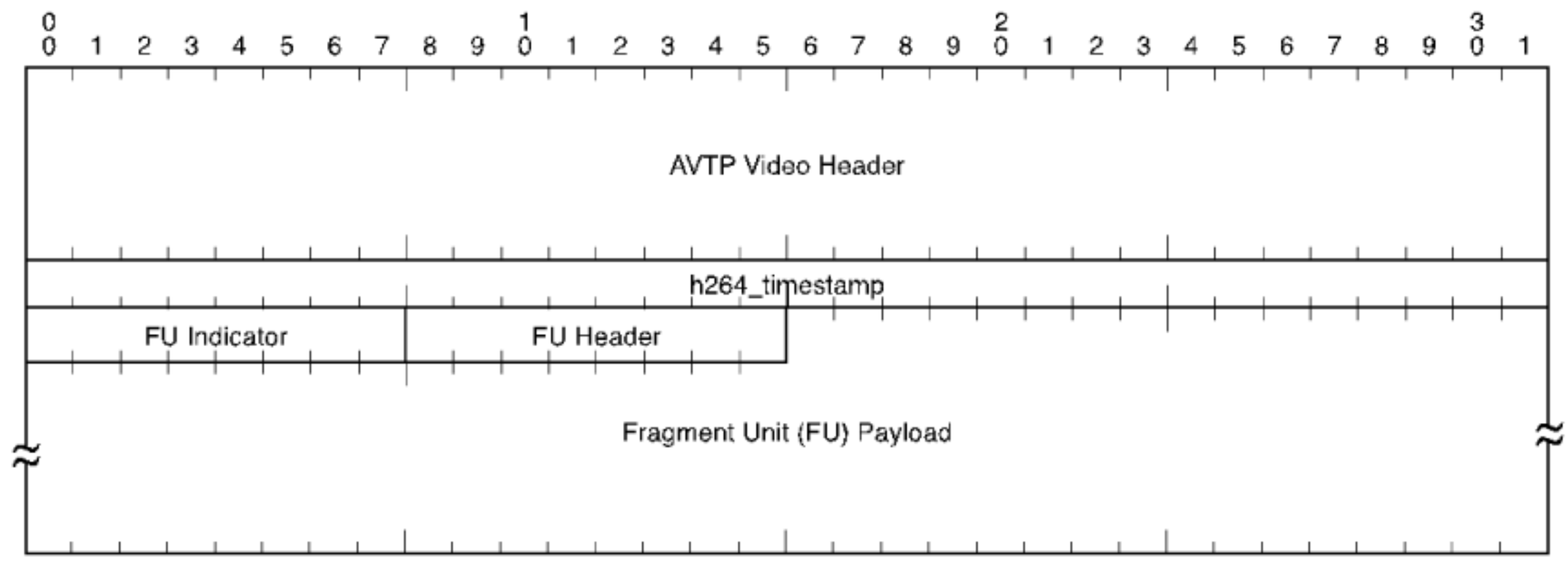


Figure 45—AVTP packet including a fragmented unit of type A

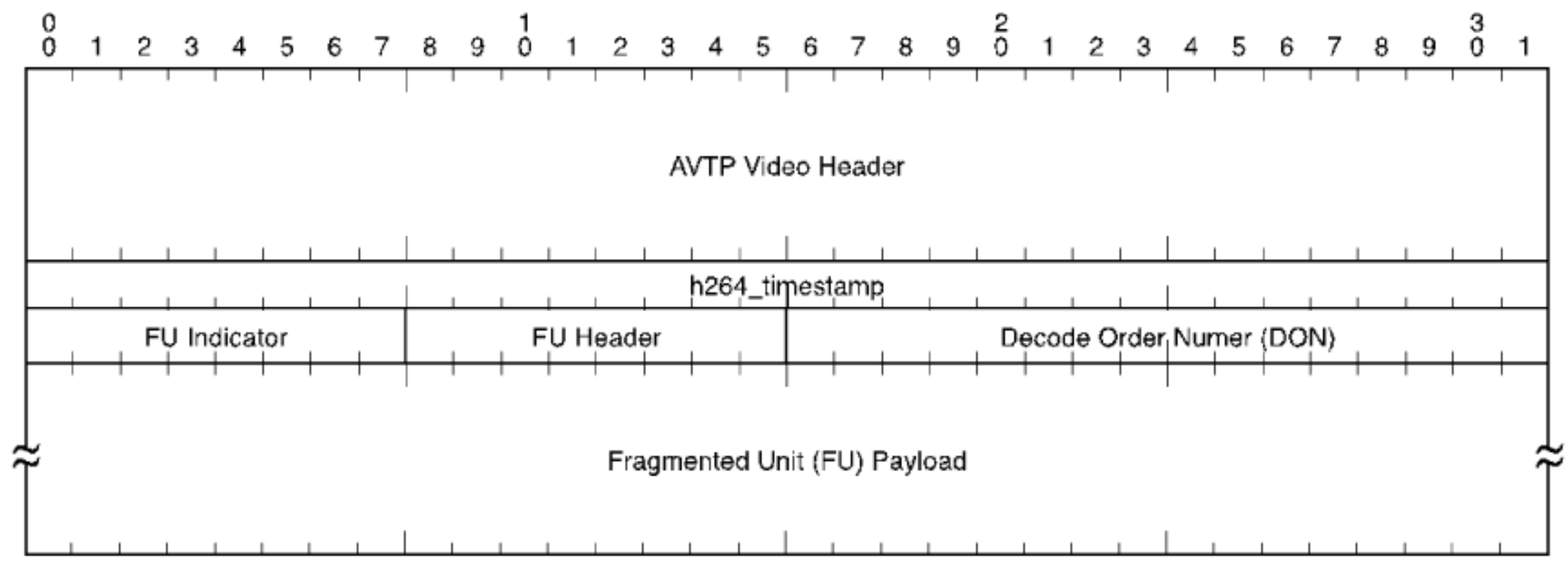


Figure 46—AVTP packet including a fragmented unit of type B

8.6 JPEG 2000 Compressed Video Format

8.6.1 Overview

The JPEG 2000 Compressed Video Format uses a **format** value of RFC (see Table 19) and a **format_subtype** value of JPEG2000 (see Table 20).

This format uses the **video_data_payload** field to define the following fields:

— **tp**: 2 bits

— **MHF**: 2 bits

— **mh_id**: 3 bits

— **T**: 1 bit

— **priority**: 1 octet

— **tile_number**: 2 octets

— **reserved**: 1 octet

Copyright © 2016 IEEE. All rights reserved.

— **fragment_offset**: 3 octets

— **JPEG_2000_payload**: 0 to *n* octets

The fields of the JPEG 2000 Compressed Video Format are given in Figure 47.

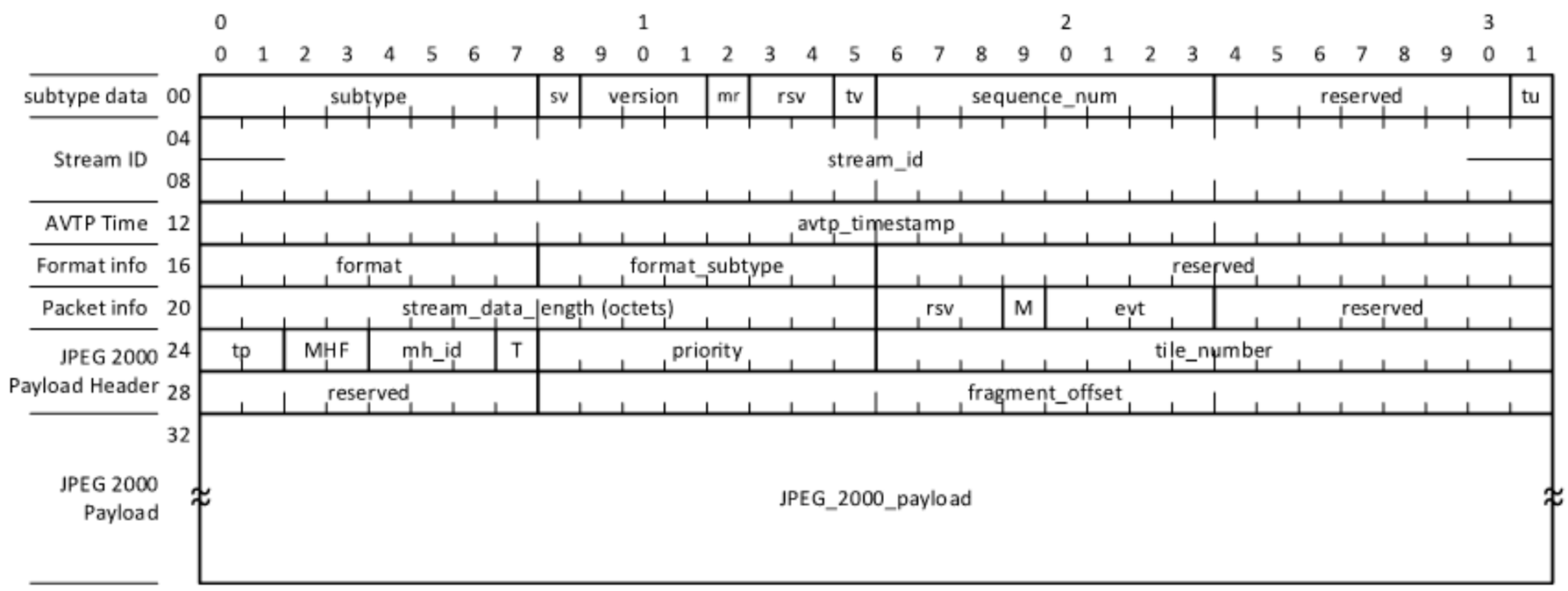


Figure 47—JPEG 2000 Compressed Video Format PDU

For each JPEG2000 AVTPDU, the AVTP header as specified in Figure 38 is followed by the JPEG 2000 payload header, which is followed by the payload, a piece of a JPEG 2000 code-stream, which is usually a JPEG 2000 packet.

8.6.2 Common header encapsulation

8.6.2.1 M field

The **M** bit of the AVTP header shall be set to one (1) for the last JPEG2000 AVTPDU of a video frame; otherwise, it shall be zero (0). When transmission is performed by multiple AVTP sessions, this bit shall be set to one (1) in the last packet of the video frame in each session.

8.6.2.2 avtp_timestamp field

The **avtp_timestamp** represents the AVTP Presentation Time associated with the given JPEG 2000 video packetization unit. It is based on the gPTP time at which the Talker receives the video packetization unit and is calculated according to the techniques described in 4.4.4.9. The Talker shall fill the **avtp_timestamp** with this AVTP Presentation Time, but only in the JPEG2000 AVTPDU carrying the final fragment of the video packetization unit. JPEG2000 AVTPDUs carrying pre-final video packetization unit fragments shall be marked with an invalid **avtp_timestamp** by setting the timestamp_valid field to zero (**tv** = 0).

8.6.3 JPEG 2000 payload header

8.6.3.1 tp field

See RFC 5371 Section 4.2.

Copyright © 2016 IEEE. All rights reserved.

8.6.3.2 MHF field

See RFC 5371 Section 4.2.

8.6.3.3 mh_id field

See RFC 5371 Section 4.2.

8.6.3.4 T field

See RFC 5371 Section 4.2.

8.6.3.5 priority field

See RFC 5371 Section 4.2.

8.6.3.6 tile_number field

See RFC 5371 Section 4.2.

8.6.3.7 fragment_offset field

See RFC 5371 Section 4.2.

8.6.4 JPEG 2000 payload

8.6.4.1 General

See RFC 5371 Sections 3 and 5.

8.6.4.2 JPEG 2000 packetization

A Talker divides the JPEG 2000 codestream into packetization units by parsing the codestream or by utilizing information received from the encoder, and packs the packetization units into JPEG2000 AVTPDUs. A Talker can put an arbitrary number of packetization units into a JPEG2000 AVTPDU, but codestream order shall be preserved. An example of this type of JPEG2000 AVTPDU is shown in Figure 48.

Copyright © 2016 IEEE. All rights reserved.

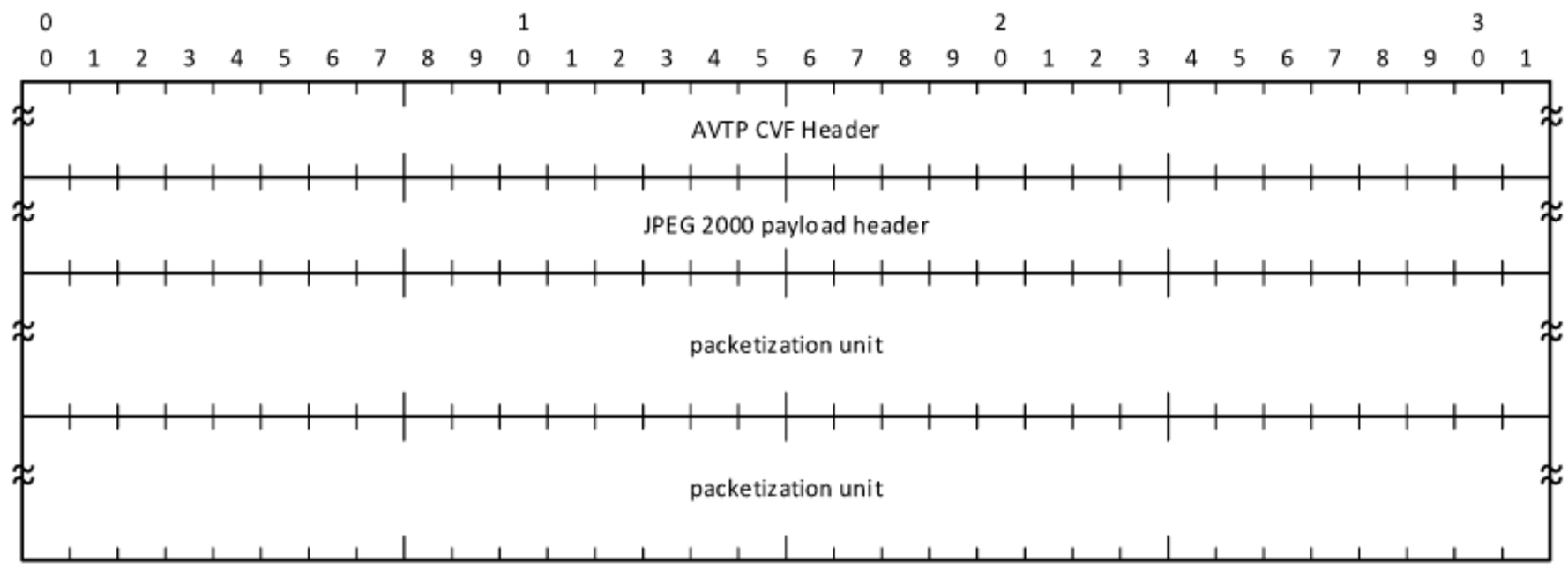


Figure 48—CVF AVTPDU with multiple packetization units

If a packetization unit with all headers is larger than the MTU size, it may be fragmented. To pack a fragmented packetization unit, each fragmented packetization unit is packed within a JPEG2000 AVTPDU. An example of this type of packetization is shown in Clause 8.

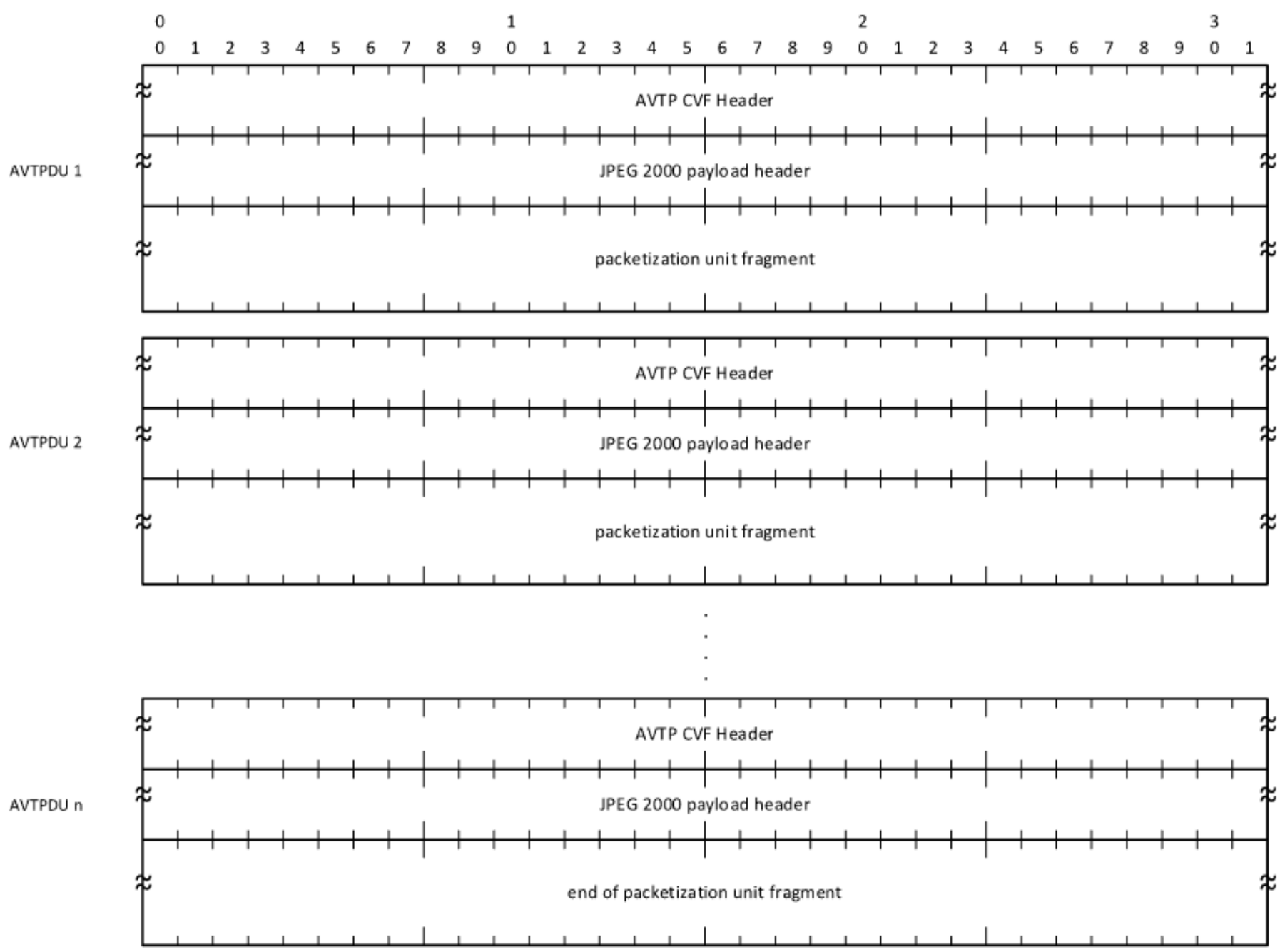


Figure 49—JPEG2000 AVTPDUs with a fragmented packetization unit

Copyright © 2016 IEEE. All rights reserved.

9. AVTP Control Format

9.1 Overview

The AVTP Control Format (ACF) provides a flexible framework for transmitting a variety of control messages in either a time-synchronous or a non–time-synchronous manner using TSN. It accomplishes this by constructing ACF AVTPDUs by combining one of two types of headers—either time-synchronous or non–time-synchronous—with a payload assembled from a concatenation of a variety of messages. ACF provides a transport alternative for control data where the reliable, bounded-, and low-latency delivery capabilities of TSN are desired.

ACF has two types of headers, time-synchronous and non–time-synchronous. The time-synchronous header is based on the common stream header (see 4.4.4) and provides an AVTP Presentation Timestamp for delivering the ACF AVTPDU to the receiving application or process at the Listener. This enables synchronous delivery of control messages and is useful in applications where synchronous behavior is required. An example is an entertainment event where lights, sound, video, pyrotechnic, and other components of the event need synchronization.

The non–time-synchronous header utilizes a subset of the fields in the common header format and common control header. It is useful for delivering messages as efficiently and quickly as possible and where synchronization of control is not required. An example is the delivery of a stream of control messages from an automotive CAN bus, across an Ethernet "backbone," to a diagnostic tool with the lowest latency and bandwidth as possible.

The payload of an ACF AVTPDU is assembled by concatenating one or more arbitrary types of messages. The individual messages in the ACF payload concatenation come from the list shown in Table 22. All messages use a mechanism very similar to the traditional Type-Length-Value (TLV)[3] structure that allows a Listener to parse the concatenated messages, even if the Listener does not recognize an individual message type. In the case where a Listener does not recognize a particular message type, it can use the length value to "jump over" the message and move on to the next message in the payload. In this way, any combination of messages is possible in the ACF AVTPDU, within the limits of the transmission medium's MTU.

There is no mechanism provided for an ACF payload to span multiple ACF AVTPDUs. Therefore, the maximum size of any ACF payload is limited to the data space available in a single ACF AVTPDU.

9.1.1 Usage notes

ACF has a multitude of uses. Please note the following guidelines and observations in the use of the ACF Format.

— Control streams are often low bit rate and therefore don't require the bandwidth held in even the minimum SRP reservation for standard TSN traffic classes. In these cases an option to consider is sending the ACF AVTPDUs as best-effort traffic and not use a formal stream reservation. This allows room for other stream reservations that need the reserved bandwidth.

— In the case where ACF AVTPDUs are sent as best-effort traffic, care should be taken with respect to addressing the Ethernet frames. Multicast addressing of best-effort traffic can flood the network. Users should consider managing the impact of multicast flooding by using techniques like grouping traffic onto separate VLANs or confining multicast traffic by utilizing MMRP, or simply using unicast addressing.

[3]The value of Length used for ACF messages has a slightly different definition from that found in traditional TLV implementations. See 9.4 for details.

Copyright © 2016 IEEE. All rights reserved.

— The technique of concatenating any number of arbitrary ACF messages to form a complete ACF payload is very flexible and provides very useful capabilities. In particular, in the case of an automotive CAN-to-Ethernet gateway, CAN messages can be concatenated from different physical CAN buses into the same stream for transport over the Ethernet network. Thus traffic from multiple CAN buses can be carried on a single ACF stream.

9.2 Non–Time-Synchronous Control Format header

9.2.1 Overview

The Non–Time-Synchronous Control Format (NTSCF) encapsulation shall use the alternative header and shall have a **subtype** field of NTSCF (see 4.4.3.2).

This encapsulation uses the following alternative header fields. See 4.4.6 for definitions.

— **version**: 3 bits

This encapsulation uses the **h** and **alternative_data_payload** fields to define the following format-specific fields:

— **sv**: 1 bit

— **r**: 1 bit

— **ntscf_data_length**: 11 bits

— **sequence_num**: 1 octet

— **stream_id**: 8 octets

— **acf_payload_data**: 0 to *n* quadlets

These fields are shown in Figure 50.

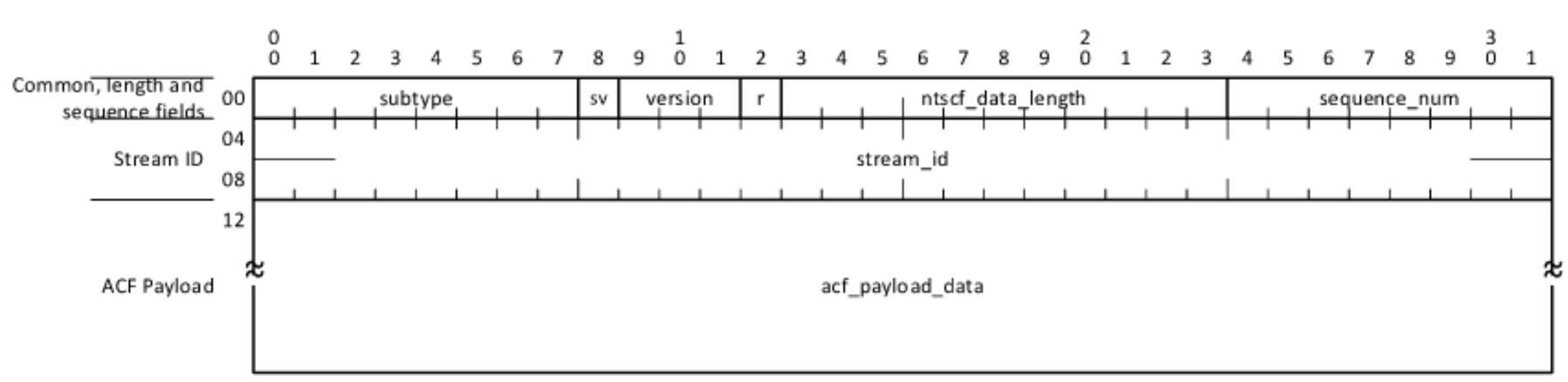


Figure 50 —Non–Time-Synchronous Control Stream PDU Format

9.2.2 sv (stream_id valid) field

The **sv** (stream_id valid) field is used to indicate whether the **stream_id** field contains a stream_id for a stream with a valid stream reservation. The options for the value of the **sv** field are defined in Table 21.

Copyright © 2016 IEEE. All rights reserved.

Table 21 —sv field values

Value	Description
0_{16}	The **stream_id** field contains a stream ID for a stream without a reservation.
1_{16}	The **stream_id** field contains a stream ID for a stream with a reservation.

See 9.2.5 for details.

9.2.3 ntscf_data_length field

The 11-bit **ntscf_data_length** field contains the length (in octets) of the **acf_data_payload** field.

The maximum value for this field is determined by the maximum transmission unit (MTU).

9.2.4 sequence_num field

The **sequence_num** field indicates the sequence of ACF AVTPDUs in a stream. This field may start at any value for the first ACF AVTPDU of a stream and shall be incremented by one (1), with wrapping, on each subsequent ACF AVTPDU of the same stream. This field is to wrap from 11111111_2 to 00000000_2 (FF_{16} to 00_{16}).

The value of this field is determined by the Talker. Listeners can use the sequence number to detect ACF AVTPDUs lost in transit.

A Listener may join a stream at any time after the stream has started and therefore, a Listener shall be able to tolerate a starting **sequence_num** of any value.

9.2.5 stream_id field

The **stream_id** field is used for stream identification. The **stream_id** field shall contain a valid StreamID as defined in subclause 35.2.2.8.2 of IEEE Std 802.1Q-2014.

NOTE—In the case where **sv** = zero (0) and there is no stream reservation associated with this ACF AVTPDU, the network will deliver the Ethernet frames for this stream as best-effort traffic. For further discussion, see Annex C.

9.2.6 acf_payload_data field

The **acf_payload_data** field is composed of one or more ACF messages using the ACF message structure described in 9.4.

9.3 Time-Synchronous Control Format header

9.3.1.1 Overview

The Time-Synchronous Control Format (TSCF) encapsulation shall use the common stream header and shall have a **subtype** field of TSCF (see 4.4.3.2).

Copyright © 2016 IEEE. All rights reserved.

This encapsulation uses the following common stream fields (see 4.4.3 and 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

— **stream_id**: 8 octets

— **avtp_timestamp**: 4 octets

— **stream_data_length**: 2 octets

This encapsulation uses the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, **format_specific_data_3**, and **stream_data_payload** fields of the common stream header to define the following format-specific fields:

— **rsv** (reserved): 2 bits

— **reserved**: 7 bits

— **reserved**: 4 octets

— **reserved**: 2 octets

— **acf_payload_data**: 0 to *n* quadlets

These fields are shown in Figure 51.

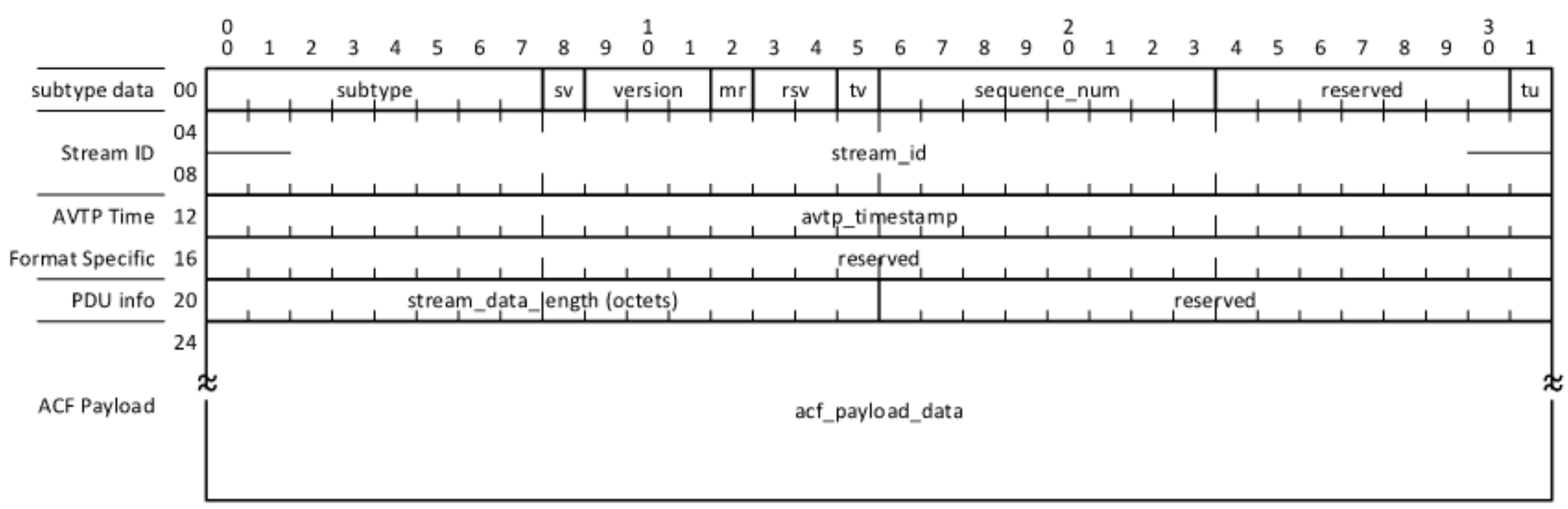


Figure 51 —Time-Synchronous Control PDU Format

9.3.2 acf_payload_data field

The **acf_payload_data** field is composed of one or more ACF messages using the ACF message structure described in 9.4.

Copyright © 2016 IEEE. All rights reserved.

9.4 ACF messages

9.4.1 Common ACF message structure

9.4.1.1 Overview

All ACF payloads are a concatenation of one or more arbitrary ACF messages. The fields in the ACF message are:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

— **acf_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 52.

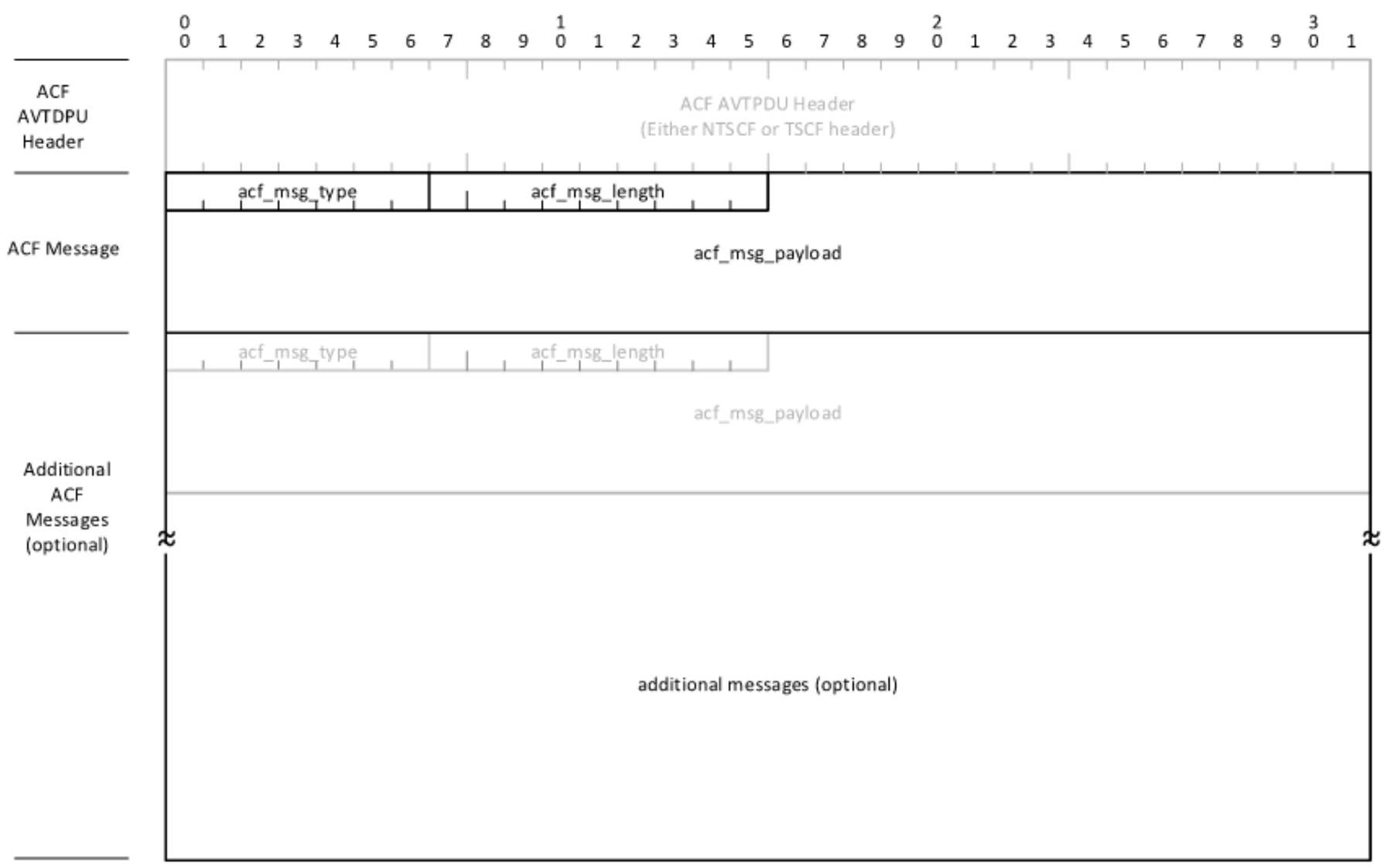


Figure 52—ACF common message structure

9.4.1.2 acf_msg_type field

The **acf_msg_type** field describes the type of message contained in the message payload. ACF message types are defined in Table 22. An ACF AVTPDU payload may contain any number of messages in any combination of types, within the constraints of a single frame.

Copyright © 2016 IEEE. All rights reserved.

Table 22—ACF message types

Value	Name	Description	Subclause
00_{16}	ACF_FLEXRAY	FlexRay™ message	9.4.2
01_{16}	ACF_CAN	Controller Area Network (CAN)/CAN with Flexible Data-Rate (CAN FD) message	9.4.3
02_{16}	ACF_CAN_BRIEF	Abbreviated CAN/CAN FD message	9.4.4
03_{16}	ACF_LIN	LIN® message	9.4.5
04_{16}	ACF_MOST	MOST® message	9.4.6
05_{16}	ACF_GPC	General purpose control message	9.4.7
06_{16}	ACF_SERIAL	Serial port message	9.4.8
07_{16}	ACF_PARALLEL	Parallel port message	9.4.9
08_{16}	ACF_SENSOR	Analog sensor message	9.4.10
09_{16}	ACF_SENSOR_BRIEF	Abbreviated sensor message	9.4.11
$0A_{16}$	ACF_AECP	IEEE Std 1722.1 AECP message	9.4.12
$0B_{16}$	ACF_ANCILLARY	Video ancillary data message	9.4.13
0C to 77_{16}	Reserved	Reserved	—
78 to $7F_{16}$	ACF_USER	User-defined ACF message	—

9.4.1.3 acf_msg_length field

The **acf_msg_length** field is an unsigned integer that indicates the number of quadlets of data contained in this message. The value of the **acf_msg_length** field includes the quadlet containing the **acf_msg_length** field and all subsequent quadlets through and including the last quadlet of **acf_msg_payload**.

To allow for efficient storage, movement, and parsing of the AVTPDU, all ACF messages shall have a total length (including the **acf_msg_type**, **acf_msg_length**, and **acf_msg_payload** fields) that is an integer multiple of quadlets. An ACF message shall have a minimum length of one (1) quadlet and the message shall be padded with zero (0) filled octets to reach this length. This means that **acf_msg_length** has a minimum value of one (1).

9.4.1.4 acf_msg_payload

The **acf_msg_payload** field contains the data of the ACF message. The format of the **acf_msg_payload** field is specific to each ACF message type as described in the remainder of this clause.

9.4.2 FlexRay™ message

9.4.2.1 Overview

The FlexRay ACF message provides the capability to transport FlexRay messages over a TSN network. It is useful in automotive applications. Use cases include interconnection of FlexRay buses across an Ethernet backbone, and connection of a FlexRay bus to a vehicle management system.

The FlexRay ACF message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

Copyright © 2016 IEEE. All rights reserved.

The FlexRay ACF message has an **acf_msg_type** field of ACF_FLEXRAY (see Table 22).

The FlexRay message defines the following message-specific fields:

— **pad** (padding length): 2 bits

— **mtv** (message_timestamp valid): 1 bit

— **fr_bus_id**: 5 bits

— **rsv** (reserved): 2 bits

— **chan** (source channel): 2 bits

— **str** (startup): 1 bit

— **syn** (sync): 1 bit

— **pre** (payload preamble): 1 bit

— **nfi** (null frame indicator): 1 bit

— **message_timestamp**: 8 octets

— **fr_frame_id**: 11 bits

— **reserved**: 15 bits

— **cycle**: 6 bits

— **flexray_msg_payload**: 0 to 64 quadlets

These fields are shown in Figure 53.

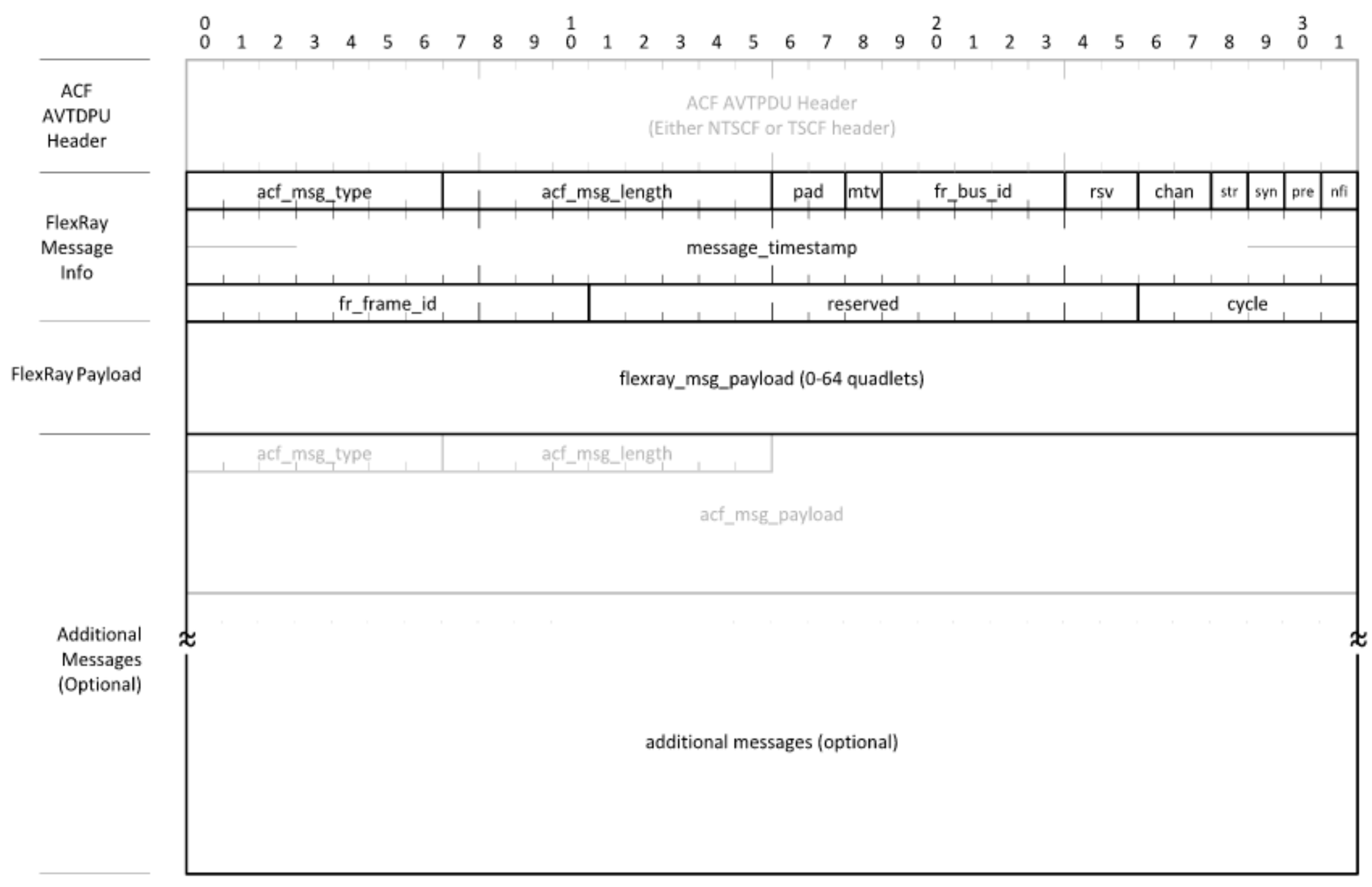


Figure 53—FlexRay ACF message

Copyright © 2016 IEEE. All rights reserved.

9.4.2.2 pad (padding length) field

The **pad** field indicates the length (in octets) of padding at the end of the message payload. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload.

9.4.2.3 mtv (message_timestamp valid) field

The **mtv** field is used to indicate whether the 8-octet **message_timestamp** field contains valid data.

The bit is set to one (1) if the **message_timestamp** field contains a valid timestamp.

The bit is set to zero (0) if the **message_timestamp** field is not valid. An AVTP Listener ignores the **message_timestamp** in any message where **mtv** is set to zero (0).

9.4.2.4 fr_bus_id field

The **fr_bus_id** field provides an identifier for the FlexRay bus on which this message originated. The value of **fr_bus_id** is application specific and is typically specified by the OEM of the vehicle. It has several potential uses, including the identification of the originating FlexRay bus when using Ethernet as a backbone to carry messages from a multiplicity of FlexRay buses in a vehicle.

If not specified for a particular application, the value of **fr_bus_id** shall be set to zero (0) by the Talker.

9.4.2.5 chan (channel) field

The **chan** bits indicate the channel on which this FlexRay message was received if the original source of the message was FlexRay. The **chan** bits are set according to Table 23.

Table 23—Source channel

Value	Description
0	The original source of the message is not a FlexRay network or is unknown.
1	The message was received on FlexRay channel A.
2	The message was received on FlexRay channel B.
3	The message was received on both FlexRay channel A and B.

While it is possible that a message may be received on both FlexRay channels, since there is only one cycle field, a message received on both channels but different cycles cannot be represented directly and is to be forwarded as two separate messages. These two messages could be transmitted in the same or in separate ACF AVTPDUs.

9.4.2.6 str (startup) field

The **str** bit contains the FlexRay message startup frame indicator.

Copyright © 2016 IEEE. All rights reserved.

9.4.2.7 syn (sync) field

The **syn** bit contains the FlexRay message sync frame indicator.

9.4.2.8 pre (preamble) field

The **pre** bit contains the FlexRay message payload preamble indicator.

9.4.2.9 nfi field

The **nfi** bit contains the FlexRay message null frame indicator.

9.4.2.10 message_timestamp field

The **message_timestamp** field contains the acquisition time in nanoseconds, if available, of the data contained in the control message. Acquisition time is the time of receipt for the FlexRay message or, if the Talker is originating this message, the timestamp is the time of creation.

The **message_timestamp** value is calculated from both the gPTP **seconds** and **nanoseconds** fields by the formula in Equation (9).

$$\text{message_timestamp} = (\text{AS_sec} \times 10^9 + \text{AS_ns}) \bmod 2^{64} \tag{9}$$

where

AS_sec the gPTP **seconds** field

AS_ns the gPTP **nanoseconds** field

If the **mtv** bit is zero (0), then the content of the **message_timestamp** field is undefined and is ignored by the ACF Listener.

9.4.2.11 fr_frame_id field

The **fr_frame_id** field contains the FlexRay message frame identifier if the message was sourced from a FlexRay network, and has a valid range of 1 to 2047. If the source of the message was not a FlexRay network then the **fr_frame_id** field is set to zero (0).

9.4.2.12 cycle field

The **cycle** field contains the FlexRay message cycle count if the message was sourced from a FlexRay network. If the source of the message was not a FlexRay network then the **cycle** field is set to zero (0).

9.4.2.13 flexray_msg_payload field

The **flexray_msg_payload** field is a variable length field that contains from 0 to 64 quadlets of message data. This capacity is sufficient to hold the largest possible FlexRay frame of 254 octets.

Copyright © 2016 IEEE. All rights reserved.

9.4.3 CAN/CAN FD message

9.4.3.1 Overview

The CAN/CAN FD ACF message provides the capability to transport CAN messages over a TSN network. It is useful in a variety of applications including automotive and staging control. Automotive use cases include interconnection of CAN buses across an Ethernet backbone and connection to an Ethernet-based diagnostic tool. Staging control system use cases include lighting and pyrotechnic control.

The CAN/CAN FD ACF message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The CAN/CAN FD ACF message has an **acf_msg_type** field of ACF_CAN (see Table 22).

The CAN message defines the following message-specific fields:

— **pad** (padding length): 2 bits

— **mtv** (message_timestamp valid): 1 bit

— **rtr** (remote transmission request): 1 bit

— **eff** (extended frame format): 1 bit

— **brs** (bit rate switch): 1 bit

— **fdf** (CAN Flexible Data-rate (FD) format): 1 bit

— **esi** (error state indicator): 1 bit

— **rsv** (reserved): 3 bits

— **can_bus_id**: 5 bits

— **message_timestamp**: 8 octets

— **rsv** (reserved): 3 bits

— **can_identifier**: 29 bits

— **can_msg_payload**: 0 to 16 quadlets

These fields are shown in Figure 54.

Copyright © 2016 IEEE. All rights reserved.

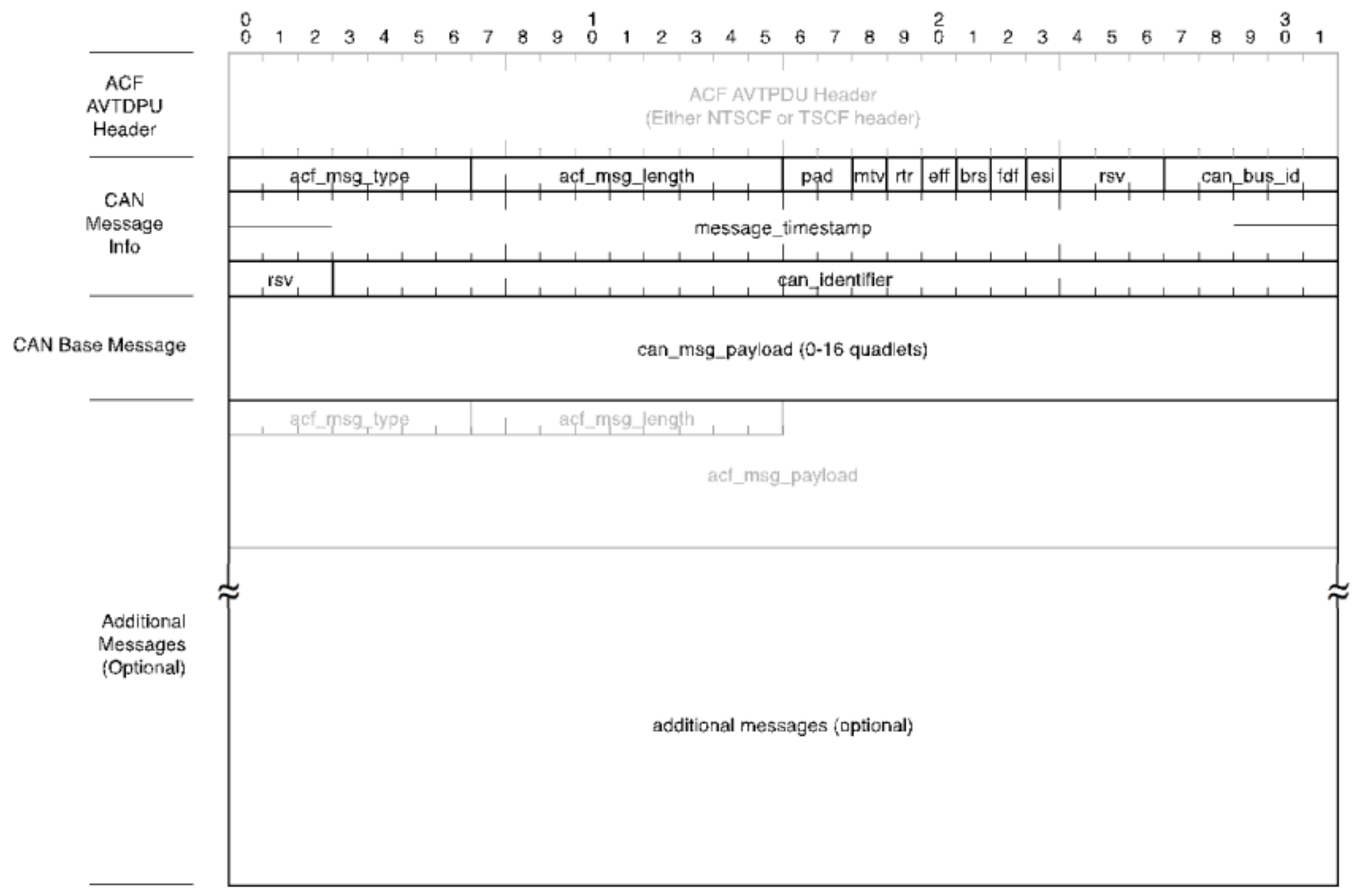


Figure 54—CAN/CAN FD ACF message

9.4.3.2 pad (padding length) field

The **pad** field indicates the length of padding at the end of the message payload in octets. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload.

NOTE—If the application requires that a CAN message be recreated from a received CAN ACF message, then the CAN payload data length (referred to as the **dlc** field in the CAN frame) can be calculated by converting **acf_msg_length** to octets and subtracting the length of the header and the padding.

9.4.3.3 mtv (message_timestamp_valid) field

The **mtv** field is used to indicate whether the 8-octet **message_timestamp** field contains valid data.

The bit is set to one (1) if the **message_timestamp** field contains a valid timestamp.

The bit is set to zero (0) if the **message_timestamp** field is not valid. An AVTP Listener ignores the **message_timestamp** in any message where **mtv** is set to zero (0).

9.4.3.4 rtr (remote_transmission_request) field

The **rtr** field contains the CAN message remote transmission request bit.

Copyright © 2016 IEEE. All rights reserved.

9.4.3.5 eff (extended_frame_format) field

The **eff** field contains the CAN message extended frame format bit.

If the **eff** bit is set to zero (0) then the **identifier** field contains an 11-bit CAN identifier.

If the **eff** bit is set to one (1) then the **identifier** field contains a 29-bit CAN identifier.

9.4.3.6 brs (bit_rate_switch) field

The **brs** field contains the CAN message bit rate switch bit.

9.4.3.7 fdf (CAN Flexible Data-rate [FD] Format) field

The **fdf** field contains the CAN message Flexible Data-rate Format bit.

If the **fdf** bit is set to zero (0) then this message is a CAN message and lengths for valid data in the **can_msg_payload** field are 0 through 8 octets.

If the **fdf** bit is set to one (1) then this message is a CAN FD message and the lengths for valid data in the **can_msg_payload** field are 0 through 8, 12, 16, 20, 24, 32, 48, and 64 octets.

9.4.3.8 esi (error_state_indicator) field

The **esi** field contains the CAN message error state indicator bit.

9.4.3.9 can_bus_id field

The **can_bus_id** field provides an identifier for the CAN bus on which this message originated. The value of **can_bus_id** is application specific and is typically specified by the OEM of the vehicle. It has several potential uses, including the identification of the originating CAN bus when using Ethernet as a backbone to carry messages from a multiplicity of CAN buses in a vehicle.

If not specified for a particular application, the value of **can_bus_id** shall be set to zero (0) by the Talker.

9.4.3.10 message_timestamp field

The **message_timestamp** field contains the acquisition time in nanoseconds, if available, of the data contained in the control message. Acquisition time is the time of receipt for the base CAN message of this CAN ACF message or, if the Talker is originating this message, the timestamp is the time of creation.

The **message_timestamp** value is calculated from both the gPTP **seconds** and **nanoseconds** fields by the formula in Equation (10).

$$\text{message_timestamp} = (\text{AS_sec} \times 10^9 + \text{AS_ns}) \bmod 2^{64} \tag{10}$$

where

AS_sec the gPTP **seconds** field

AS_ns the gPTP **nanoseconds** field

Copyright © 2016 IEEE. All rights reserved.

If the **mtv** bit is zero (0), then the content of the **message_timestamp** field is undefined and is ignored by the ACF Listener.

9.4.3.11 can_identifier field

The 29-bit **can_identifier** field contains the CAN message identifier. CAN message identifiers are either 11 or 29 bits in length. The length of the **can_identifier** field is communicated by the value of the **eff** bit (see 9.4.3.5). 11-bit CAN message identifiers are stored in bits 21 through 31 of the quadlet holding the **can_identifier** field and the remaining bits of the field are set to zero (0). 29-bit CAN message identifiers occupy the entire 29-bit field. Storage of the 11-bit CAN identifier is shown in Figure 55. The letter *v* indicates a valid identifier bit.

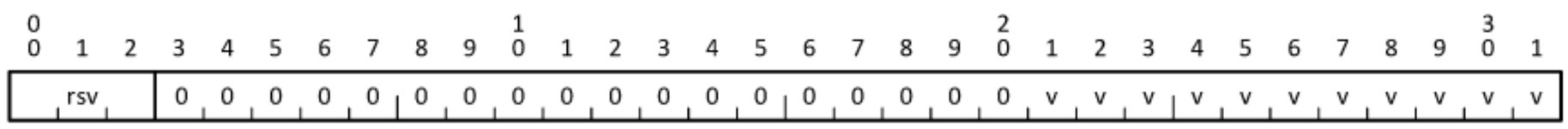


Figure 55—Storage of an 11-bit CAN identifier

9.4.3.12 can_msg_payload field

The **can_msg_payload** field is a variable length field that contains from 0 to 16 quadlets of message data. This is sufficient to hold the maximum CAN frame of 64 octets. The overall size of the **can_msg_payload** field shall always be an integer multiple of quadlets. Table 24 provides a few examples of CAN payload calculations.

Table 24—Example CAN payloads

Original CAN message length	Payload length	pad value	acf_msg_length value
0	0	0	4
7 octets	2 quadlets	1	6
32 octets	8 quadlets	0	12
64 octets	16 quadlets	0	20

9.4.4 Abbreviated CAN/CAN FD message

The Abbreviated CAN/CAN FD ACF message provides the same functionality as the CAN/CAN FD ACF message (see 9.4.3) to transport CAN messages over a TSN network, except that it removes the message timestamp. The Abbreviated CAN/CAN FD ACF message removes the message timestamp to provide more efficient transport of CAN data in scenarios where timestamping is not critical.

The Abbreviated CAN/CAN FD (CAN_BRIEF) ACF message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The CAN_BRIEF message has an **acf_msg_type** field of ACF_CAN_BRIEF (see Table 22).

Copyright © 2016 IEEE. All rights reserved.

The CAN_BRIEF message defines the following message-specific fields:

— **pad** (padding length): 2 bits

— **mtv** (message_timestamp valid): 1 bit

— **rtr** (remote transmission request): 1 bit

— **eff** (extended frame format): 1 bit

— **brs** (bit rate switch): 1 bit

— **fdf** (CAN Flexible Data-rate Format): 1 bit

— **esi** (error state indicator): 1 bit

— **rsv** (reserved): 3 bits

— **can_bus_id**: 5 bits

— **can_identifier**: 29 bits

— **rsv**: 3 bits

— **can_msg_payload**: 0 to 16 quadlets

These fields are shown in Figure 56.

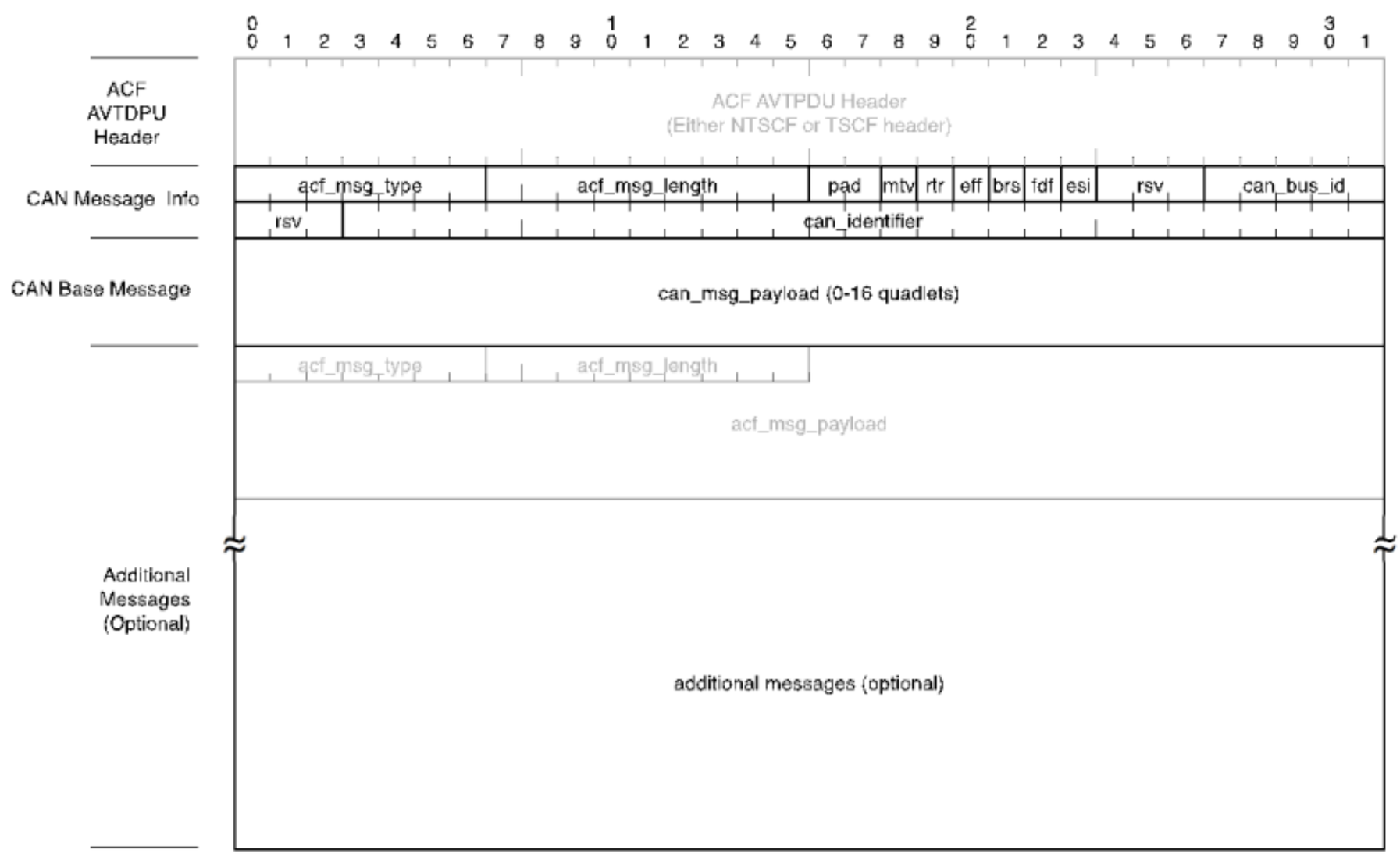


Figure 56—Abbreviated CAN/CAN FD ACF message

The CAN_BRIEF message has the same fields as the standard CAN message (9.4.3), except that it does not have the two (2) quadlet timestamp field. The CAN_BRIEF message is designed to minimize PDU size and maximize network efficiency when carrying CAN messages for which the acquisition timestamp is not necessary.

Copyright © 2016 IEEE. All rights reserved.

9.4.5 LIN® message

9.4.5.1 Overview

The LIN ACF message provides the capability to transport LIN messages over a TSN network. It is useful in automotive applications. Use cases include connection of one or more LIN buses to a central control unit within an automobile.

The LIN ACF message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The LIN message has an **acf_msg_type** field of ACF_LIN (see Table 22).

The LIN message defines the following message-specific fields:

— **pad** (padding length): 2 bits

— **mtv** (message_timestamp valid): 1 bit

— **lin_bus_id**: 5 bits

— **lin_identifier**: 1 octet

— **message_timestamp**: 8 octets

— **lin_msg_payload**: 0 to 2 quadlets

These fields are given in Figure 57.

Copyright © 2016 IEEE. All rights reserved.

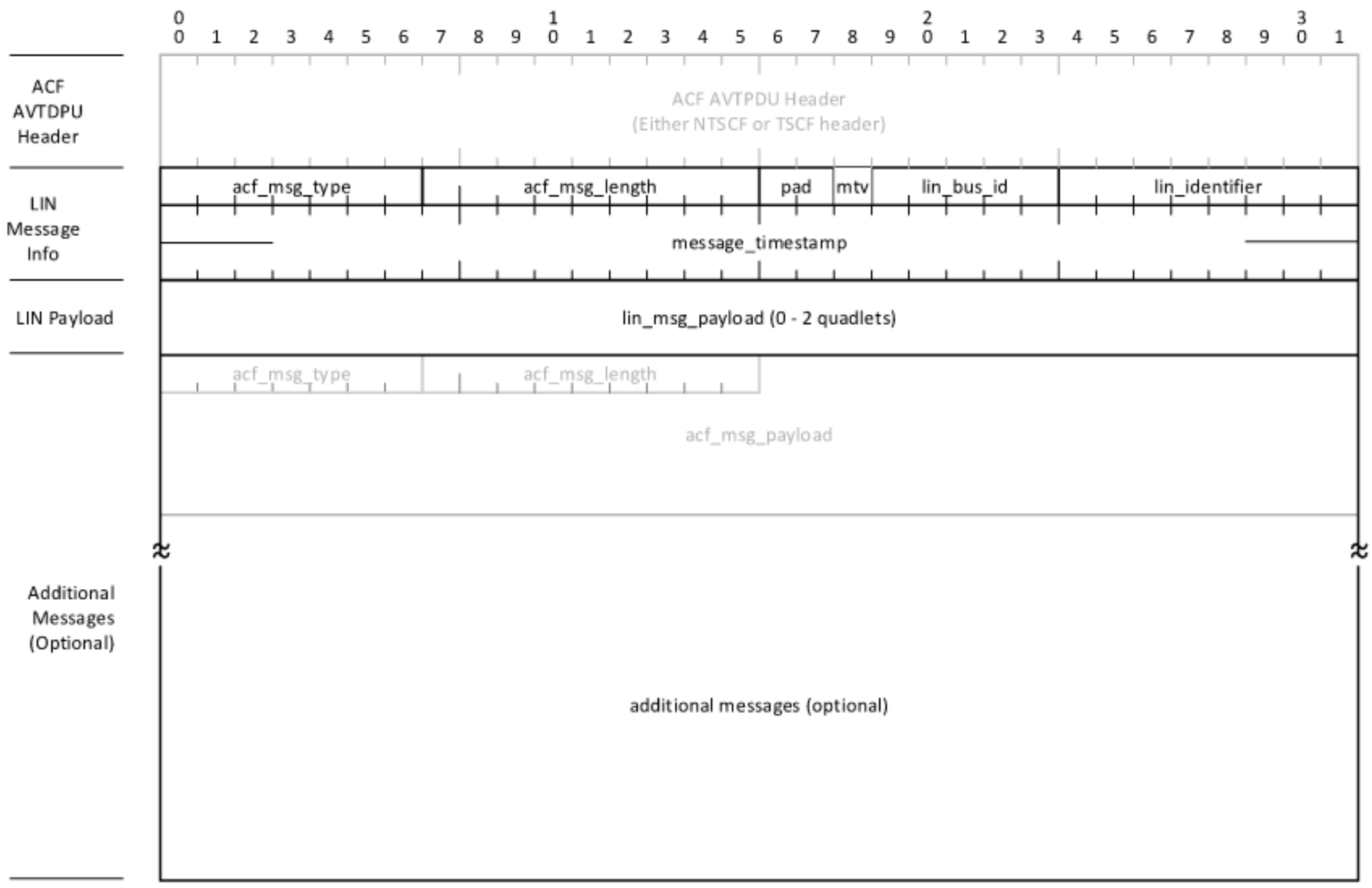


Figure 57 —LIN ACF message

9.4.5.2 pad (padding length) field

The **pad** field indicates the length of padding at the end of the message payload in octets. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload.

9.4.5.3 mtv (message_timestamp_valid) field

The **mtv** field is used to indicate whether the 8-octet **message_timestamp** field contains valid data.

The bit is set to one (1) if the **message_timestamp** field contains a valid timestamp.

The bit is set to zero (0) if the **message_timestamp** field is not valid. An AVTP Listener ignores the **message_timestamp** in any message where **mtv** is set to zero (0).

9.4.5.4 lin_bus_id field

The **lin_bus_id** field provides an identifier for the LIN bus on which this message originated. The value of **lin_bus_id** is application specific and is typically specified by the OEM of the vehicle. It has several potential uses, including the identification of the originating LIN bus when using Ethernet as a backbone to carry messages from a multiplicity of LIN buses in a vehicle.

If not specified for a particular application, the value of **lin_bus_id** shall be set to zero (0) by the Talker.

Copyright © 2016 IEEE. All rights reserved.

9.4.5.5 lin_identifier field

The 1-octet **lin_identifier** field contains the LIN message identifier.

9.4.5.6 message_timestamp field

The **message_timestamp** field contains the acquisition time in nanoseconds, if available, of the data contained in the control message. Acquisition time is the time of receipt for the base LIN message of this LIN ACF message or, if the Talker is originating this message, the timestamp is the time of creation.

The **message_timestamp** value is calculated from both the gPTP **seconds** and **nanoseconds** fields by the formula in Equation (11).

$$\text{message_timestamp} = (\text{AS_sec} \times 10^9 + \text{AS_ns}) \bmod 2^{64} \tag{11}$$

where

AS_sec the gPTP **seconds** field

AS_ns the gPTP **nanoseconds** field

If the **mtv** bit is zero (0), then the content of the **message_timestamp** field is undefined and is ignored by the ACF Listener.

9.4.5.7 lin_msg_payload field

The **lin_msg_payload** field is a variable length field that contains from 0 to 8 octets of valid message data.

9.4.6 MOST® message

9.4.6.1 Overview

The MOST ACF message provides the capability to transport MOST messages over a TSN network. It is useful in automotive applications. Use cases include interconnection of MOST buses across an Ethernet backbone, and connection of a MOST bus to a diagnostic system.

The MOST ACF message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The MOST message has an **acf_msg_type** field of ACF_MOST (see Table 22).

The MOST message defines the following message-specific fields:

— **pad** (padding length): 2 bits

— **mtv** (message_timestamp valid): 1 bit

— **most_net_id**: 5 bits

— **reserved**: 1 octet

Copyright © 2016 IEEE. All rights reserved.

— **message_timestamp**: 8 octets

— **device_id**: 2 octets

— **fblock_id**: 1 octet

— **inst_id**: 1 octet

— **func_id**: 12 bits

— **op_type**: 4 bits

— **reserved**: 2 octets

— **most_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 58.

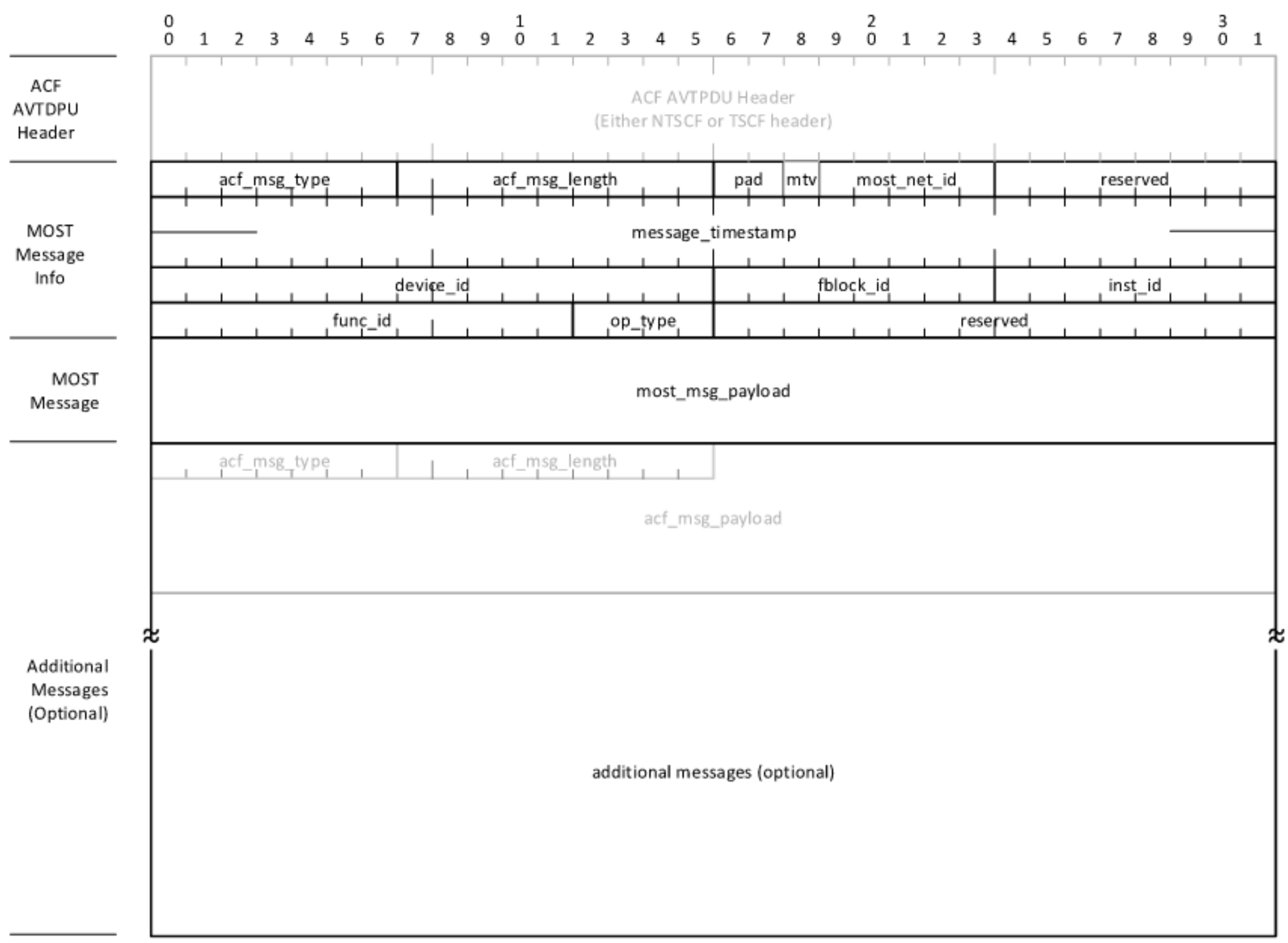


Figure 58 —MOST ACF message

9.4.6.2 pad (padding length) field

The **pad** field indicates the length of padding at the end of the message payload in octets. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload.

Copyright © 2016 IEEE. All rights reserved.

9.4.6.3 mtv (message_timestamp_valid) field

The **mtv** field is used to indicate whether the 8-octet **message_timestamp** field contains valid data.

The bit is set to one (1) if the **message_timestamp** field contains a valid timestamp.

The bit is set to zero (0) if the **message_timestamp** field is not valid. An AVTP Listener ignores the **message_timestamp** in any message where **mtv** is set to zero (0).

9.4.6.4 most_net_id field

The **most_net_id** field provides an identifier for the MOST bus on which this message originated. The value of **most_net_id** is application specific and is typically specified by the OEM of the vehicle. It has several potential uses, including the identification of the originating MOST bus when using Ethernet as a backbone to carry messages from a multiplicity of MOST buses in a vehicle.

If not specified for a particular application, the value of **most_net_id** shall be set to zero (0) by the Talker.

9.4.6.5 message_timestamp field

The **message_timestamp** field contains the acquisition time in nanoseconds, if available, of the data contained in the control message. Acquisition time is the time of receipt for the base MOST message of this MOST ACF message or, if the Talker is originating this message, the timestamp is the time of creation.

The **message_timestamp** value is calculated from both the gPTP **seconds** and **nanoseconds** fields by the formula in Equation (12).

$$\text{message_timestamp} = (\text{AS_sec} \times 10^9 + \text{AS_ns}) \bmod 2^{64} \tag{12}$$

where

AS_sec the gPTP **seconds** field

AS_ns the gPTP **nanoseconds** field

If the **mtv** bit is zero (0), then the content of the **message_timestamp** field is undefined and is ignored by the ACF Listener.

9.4.6.6 device_id field

The **device_id** field contains the node address of the source node as defined by MOST. If a function block (FBlock) receives a message, the **device_id** contains the node address of the sender. In the case of a MOST answer, the **device_id** contains the node address of the MOST receiver.

9.4.6.7 fblock_id field

The **fblock_id** field contains the identifier of a MOST FBlock.

Copyright © 2016 IEEE. All rights reserved.

9.4.6.8 inst_id field

There may be several equal FBlocks with the same **fblock_id** in a particular system. The **inst_id** provides an 8-bit unique identification number for each FBlock in each category of FBlocks. The combination of **fblock_id** and **inst_id** uniquely identifies each element in the MOST network.

9.4.6.9 func_id field

The **func_id** field indicates the MOST function for this FBlock.

9.4.6.10 op_type field

The **op_type** field indicates the MOST operation associated with this message.

9.4.6.11 most_msg_payload field

The **most_msg_payload** field contains the MOST message. For messages larger than can be accommodated in a single standard frame, a higher-level protocol is needed to provide the necessary segmentation and reassembly of the MOST message.

9.4.7 General Purpose Control (GPC) message

9.4.7.1 Overview

The General Purpose Control (GPC) ACF message provides a generic message structure. It is useful in applications where another ACF message is not suitable and a user needs to construct a specialized message for their application.

The GPC message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The GPC message has an **acf_msg_type** field of ACF_GPC (see Table 22).

The GPC message defines the following message-specific fields:

— **gpc_msg_id**: 6-octets

— **gpc_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 59.

Copyright © 2016 IEEE. All rights reserved.

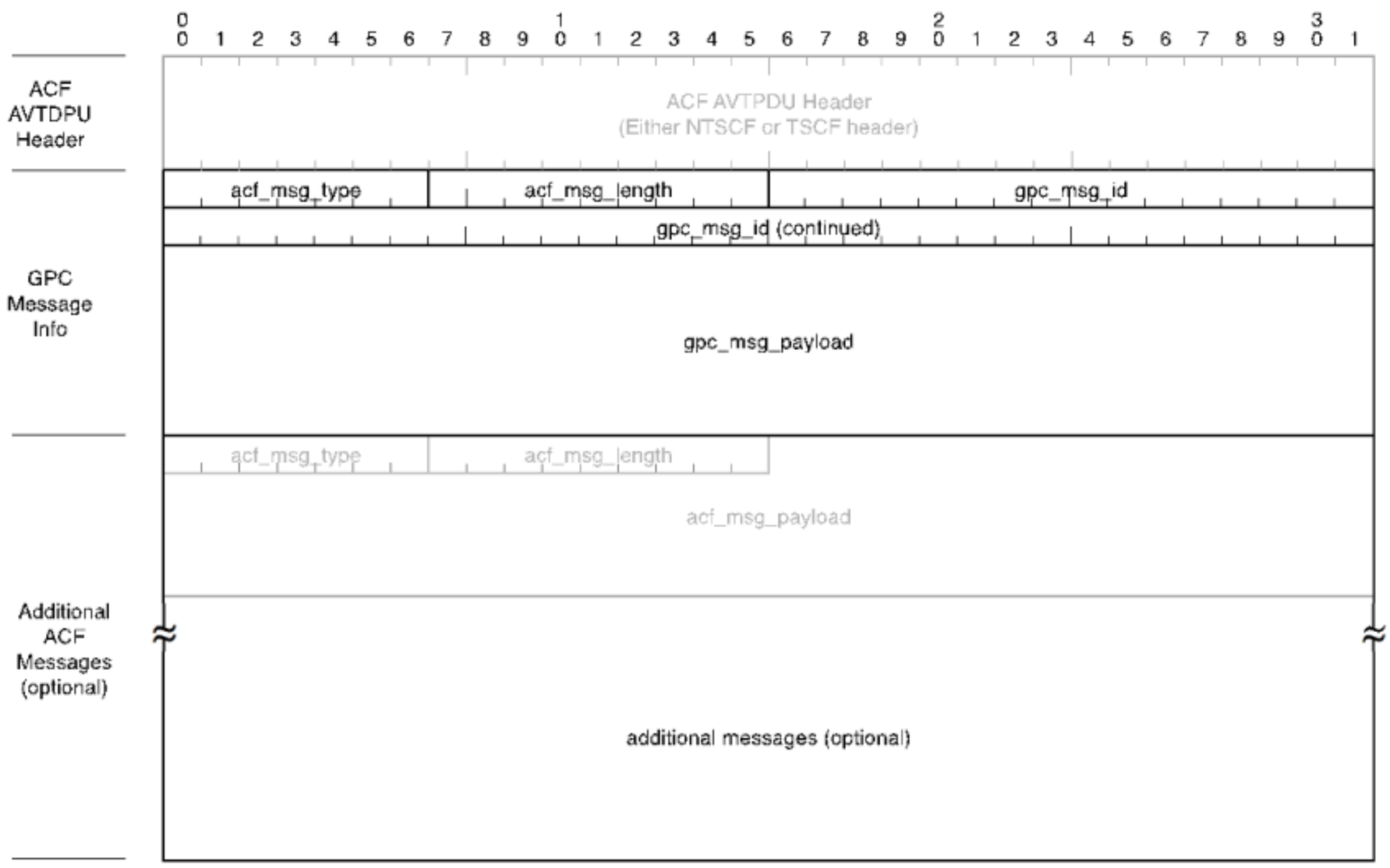


Figure 59—General Purpose Control (GPC) message

9.4.7.2 gpc_msg_id field

The **gpc_msg_id** field is a 48-bit (6-octet) field identifying the specific GPC message. The **gpc_msg_id** provides a unique identifier for a GPC message type that is defined by the organization creating the message type. It can be constructed in two ways. The first octets contain the organization's MA-L (OUI), MA-M, or MA-S, followed by a unique value that fills out the 48-bit field assigned by the organization holding the identifier. Alternatively, the first octets contain the organization's CID, followed by a unique value that fills out the 48-bit field assigned by the organization holding the CID.[4]

9.4.7.3 gpc_msg_payload field

The **gpc_msg_payload** field contains the payload of the GPC message. The structure of the payload is defined by the value of the **gpc_msg_id** field.

9.4.8 Serial (SERIAL) message

9.4.8.1 Overview

The Serial (SERIAL) ACF message provides the capability to transport data to/from a RS-232 serial interface over a TSN network. It is useful in control applications. Use cases include consolidation and transmission of multiple RS-232 control interfaces over a single Ethernet network to a common control system.

[4] https://standards.ieee.org/develop/regauth/tut/eui.pdf provides information on OUIs and CIDs.

Copyright © 2016 IEEE. All rights reserved.

The Serial (SERIAL) message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The SERIAL message has an **acf_msg_type** field of ACF_SERIAL (see Table 22).

The SERIAL message defines the following message-specific fields:

— **pad** (padding length): 2 bits

— **reserved**: 1 octet

— **dcd** (data carrier detect): 1 bit

— **dtr** (data terminal ready): 1 bit

— **dsr** (data set ready): 1 bit

— **rts** (request to send): 1 bit

— **cts** (clear to send): 1 bit

— **ri** (ring indicator): 1 bit

— **serial_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 60.

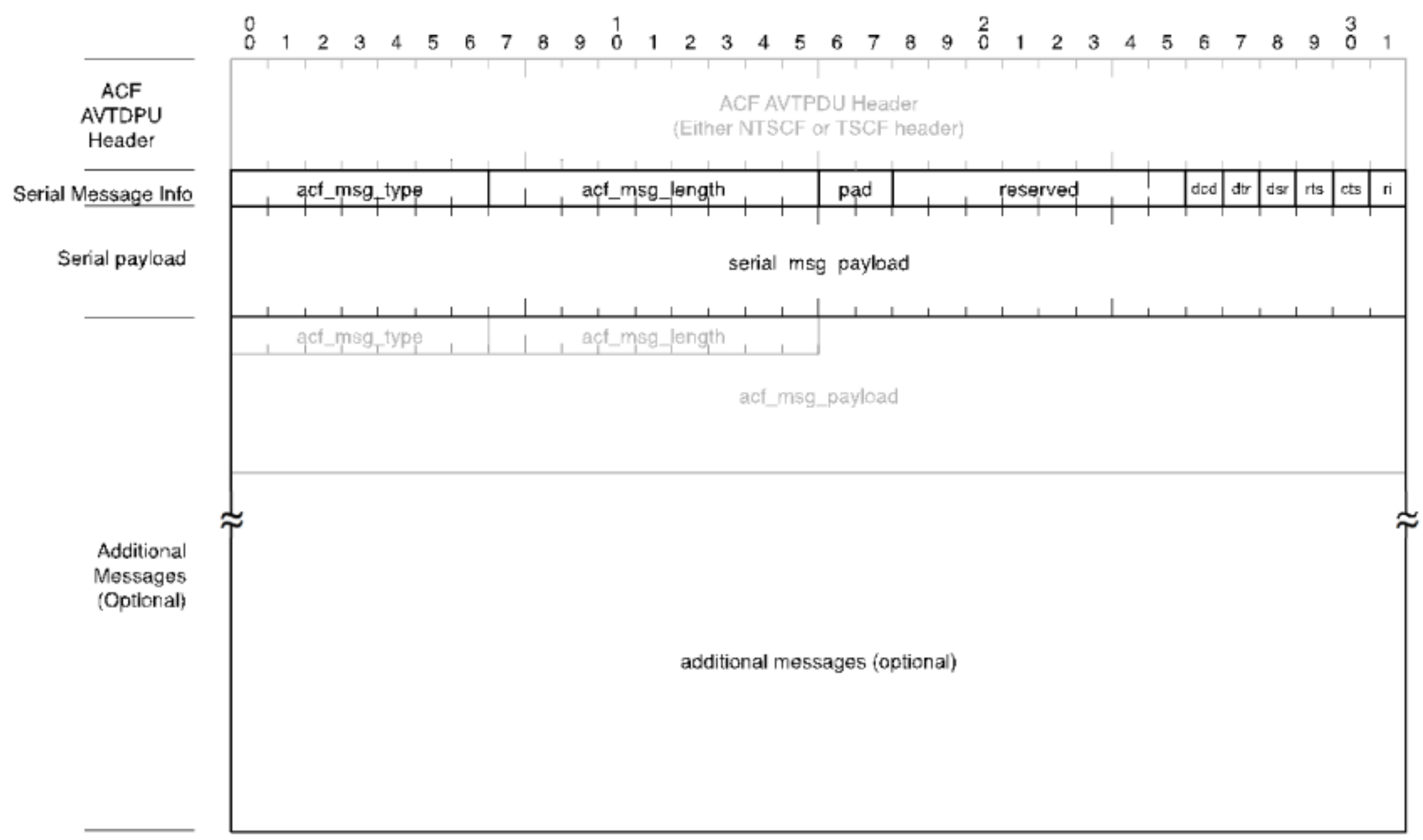


Figure 60 —Serial (SERIAL) message

Copyright © 2016 IEEE. All rights reserved.

9.4.8.2 pad (padding length) field

The **pad** field indicates the length of padding at the end of the message payload in octets. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload.

9.4.8.3 dcd (data carrier detect) field

The **dcd** field contains the status of the DCD pin of the RS-232C port when applicable.

9.4.8.4 dtr (data terminal ready) field

The **dtr** field contains the status of the DTR pin of the RS-232C port when applicable.

9.4.8.5 dsr (data set ready) field

The **dsr** field contains the status of the DSR pin of the RS-232C port when applicable.

9.4.8.6 rts (request to send) field

The **rts** field contains the status of the RTS pin of the RS-232C port when applicable.

9.4.8.7 cts (clear to send) field

The **cts** field contains the status of the CTS pin of the RS-232C port when applicable.

9.4.8.8 ri (ring indicator) field

The **ri** field contains the status of the RI pin of the RS-232C port when applicable.

9.4.8.9 serial_msg_payload field

The **serial_msg_payload** field contains the payload of the serial message.

9.4.9 Parallel (PARALLEL) message

9.4.9.1 Overview

The Parallel (PARALLEL) ACF message provides the capability to transport data to/from a parallel interface, or to an array of binary control points. It is useful in monitoring and operating binary control points. Use cases include a control system that is monitoring a multitude of switch closures in an industrial control application.

Copyright © 2016 IEEE. All rights reserved.

The Parallel (PARALLEL) message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The PARALLEL message has an **acf_msg_type** field of ACF_PARALLEL (see Table 22).

The PARALLEL message defines the following message-specific fields:

— **reserved**: 1 octet

— **bit_width**: 1 octet

— **parallel_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 61.

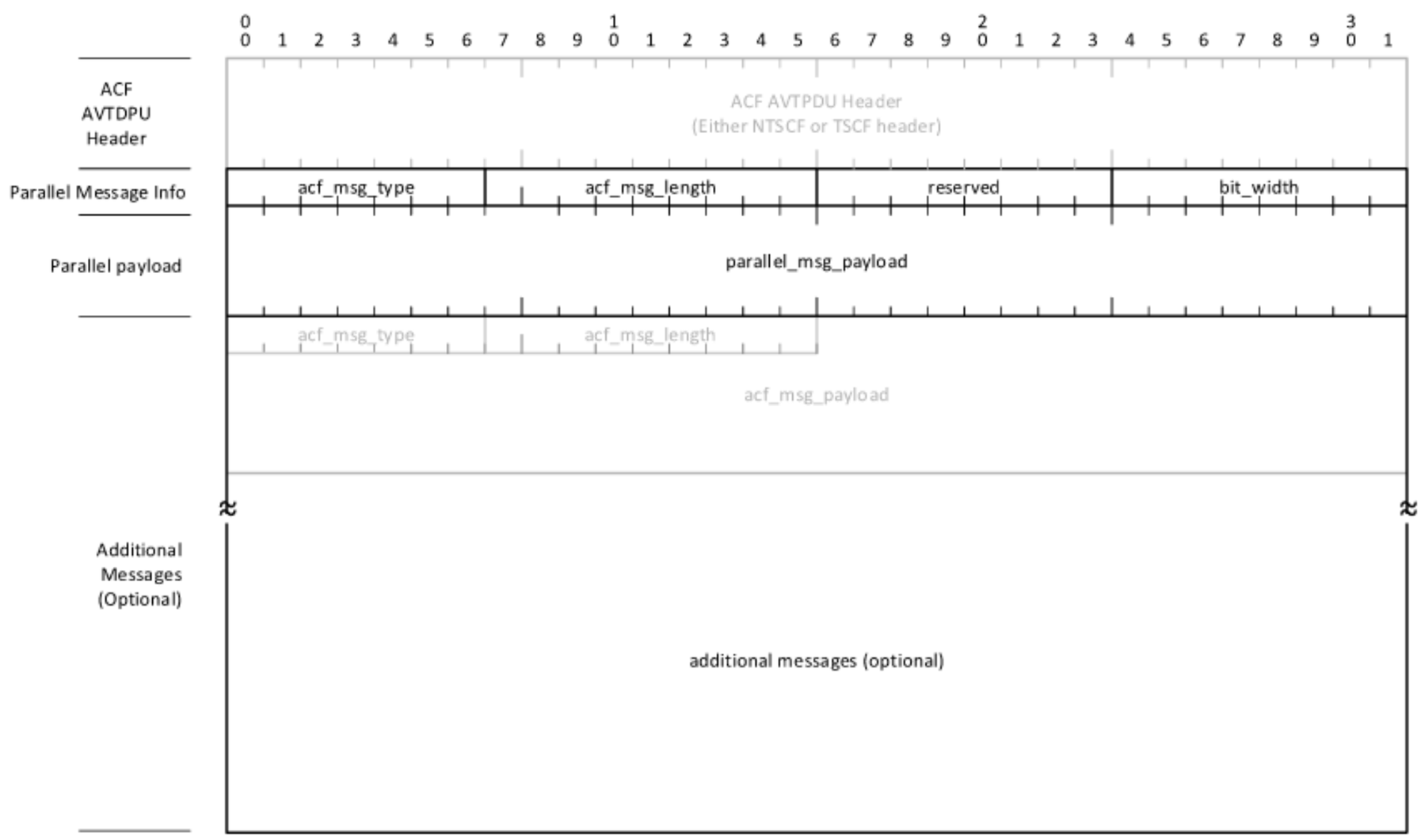


Figure 61 —Parallel (PARALLEL) message

9.4.9.2 bit_width field

The **bit_width** field contains the number of valid bits in the **parallel_msg_payload** field starting at bit 0. When the **bit_width** field is zero (0), the number of valid bits in the **parallel_msg_payload** field is 256. **parallel_msg_payload** is zero-padded to the nearest quadlet boundary and **acf_msg_length** is set according to 9.4.1.

Copyright © 2016 IEEE. All rights reserved.

9.4.9.3 parallel_msg_payload field

The **parallel_msg_payload** field contains the payload of the parallel message. Each bit position in the **parallel_msg_payload** carries the value of the corresponding pin on the parallel interface.

Figure 62 shows an example payload when **bit_width** = 67. The letter *v* in the figure indicates that a given bit position carries valid data for the corresponding pin on parallel interface. In this case **acf_msg_length** has a value of four (4).

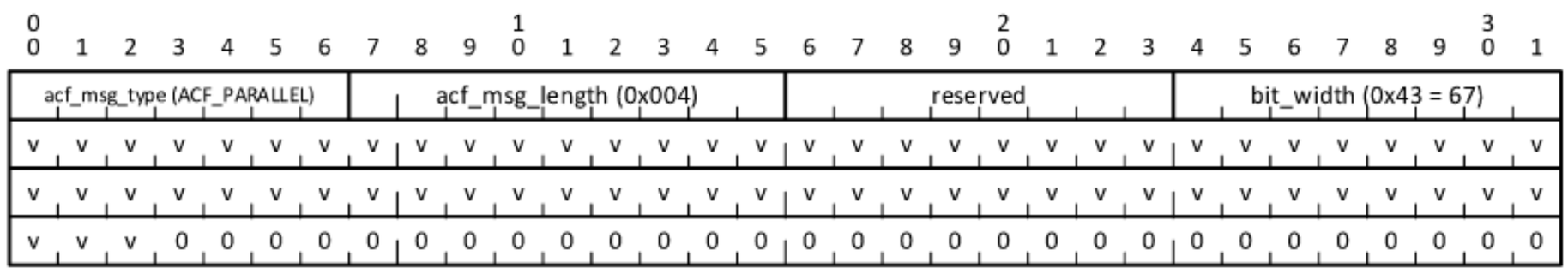


Figure 62—Example Parallel message with bit_width = 67

9.4.10 Sensor (SENSOR) message

9.4.10.1 Overview

The Sensor (SENSOR) ACF message provides the capability to transport data to/from an array of sensors. It is useful in system-control applications using real-time sensors. Use cases include engine control, where the SENSOR message is used to carry engine sensor data to a central controller, or robotic control where the SENSOR message is used to carry positional information to a central movement controller.

The Sensor (SENSOR) message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The SENSOR message has an **acf_msg_type** field of ACF_SENSOR (see Table 22).

The SENSOR message defines the following message-specific fields:

— **mtv** (message_timestamp valid): 1 bit

— **num_sensors**: 7 bits

— **sz** (sensor value size): 2 bits

— **sensor_group**: 6 bits

— **sensor_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 63.

Copyright © 2016 IEEE. All rights reserved.

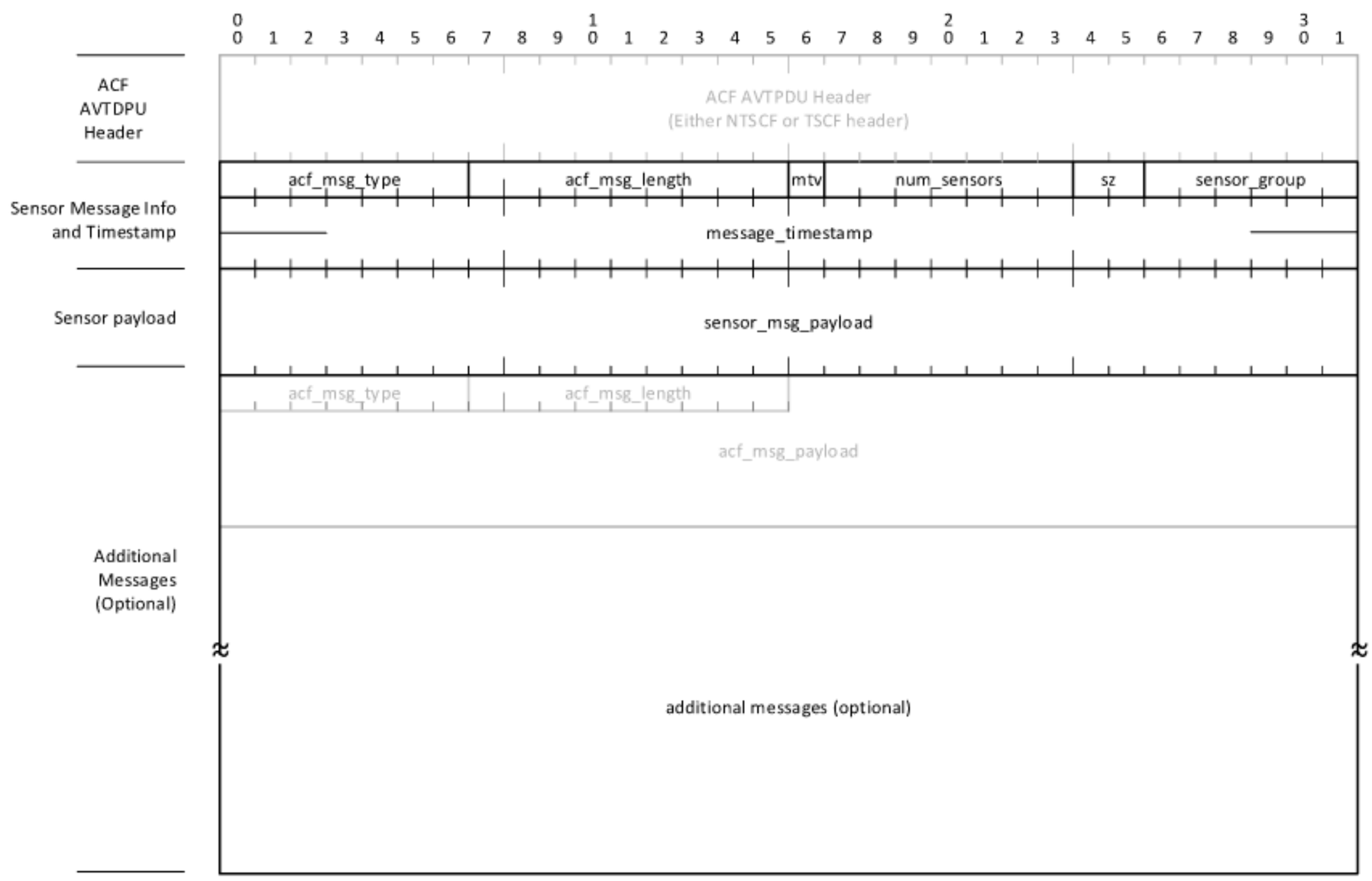


Figure 63—Sensor (SENSOR) message

9.4.10.2 mtv (message_timestamp_valid) field

The **mtv** field is used to indicate whether the 8-octet **message_timestamp** field contains valid data.

The bit is set to one (1) if the **message_timestamp** field contains a valid timestamp.

The bit is set to zero (0) if the **message_timestamp** field is not valid. An AVTP Listener ignores the **message_timestamp** in any message where **mtv** is set to zero (0).

9.4.10.3 num_sensors field

The **num_sensors** field contains the number of sensor values in the **sensor_msg_payload** field. When the **num_sensors** field is zero (0), the number of sensor values in the **sensor_msg_payload** field is 128.

9.4.10.4 sz (sensor value size) field

The **sz** field specifies the size of each sensor data value in octets. When the **sz** field is zero (0), each sensor value is 4 octets in length.

9.4.10.5 sensor_group field

The **sensor_group** field carries an application-specific identifier for a grouping of sensors. As an example, the **sensor_group** field may be used to identify a set of sensors with common characteristics for a

Copyright © 2016 IEEE. All rights reserved.

particular application. The assignment of **sensor_group** values and the assembly of sensor groups to form an entire sensor array is beyond the scope of this document.

9.4.10.6 message_timestamp field

The **message_timestamp** field contains the acquisition time in nanoseconds, if available, of the data contained in the control message.

The **message_timestamp** value is calculated in the same way as in the ACF_CAN message (see 9.4.3).

If the **mtv** bit is zero (0), then the content of the **message_timestamp** field is undefined and is ignored by the ACF Listener.

9.4.10.7 sensor_msg_payload field

The **sensor_msg_payload** field contains the data values for the sensors. Sensor data is arranged sequentially starting with the data value for sensor[0] and increasing to the data value for sensor[**num_sensors** − 1]. The **sensor_msg_payload** field shall be zero-padded as described in 9.4.1.

Figure 64 shows an example of a payload with 17 1-octet sensor values. In this case **acf_msg_length** = 8.

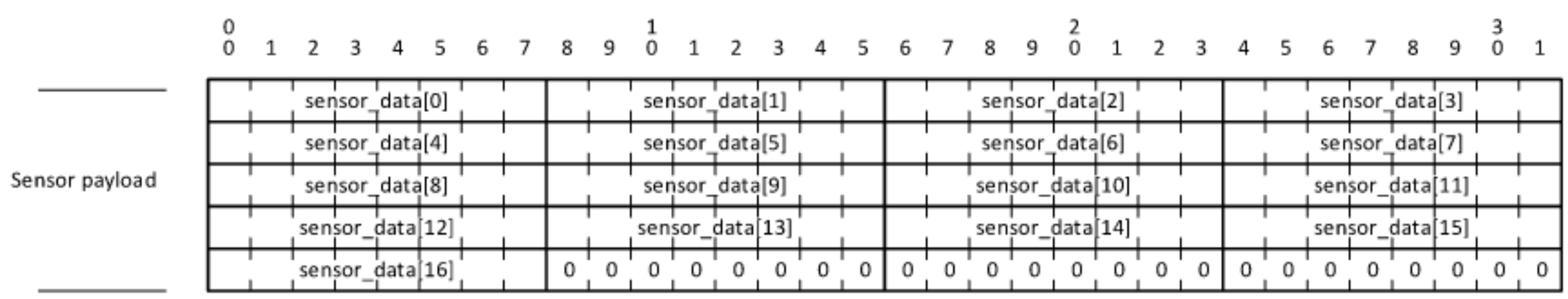


Figure 64—Example Sensor message payload with num_sensors = 17 and sz = 1

Figure 65 shows an example of a payload with 19 2-octet sensor values. In this case **acf_msg_length** = 13.

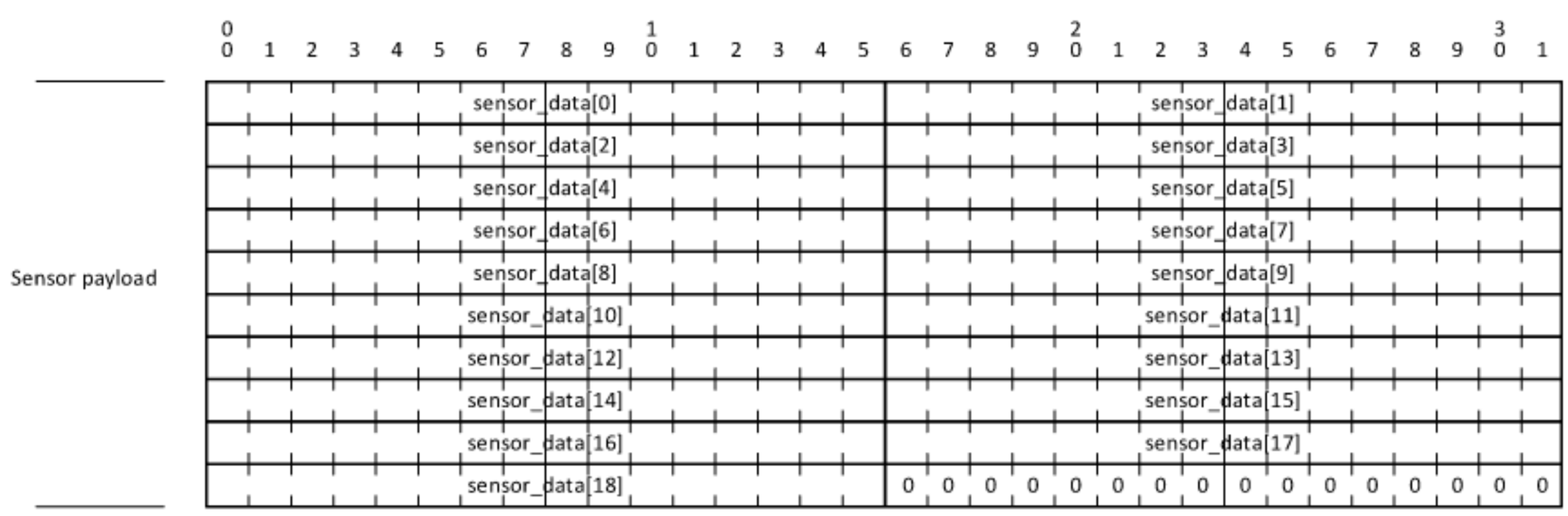


Figure 65—Example Sensor message payload with num_sensors = 19 and sz = 2

Copyright © 2016 IEEE. All rights reserved.

9.4.11 Abbreviated Sensor message

The Abbreviated Sensor ACF message provides the same capability as the Sensor ACF message (see 9.4.10) to transport data to/from an array of sensors. The Abbreviated Sensor ACF message removes the message timestamp to provide more efficient transport of sensor data in scenarios where timestamping is not critical.

The Abbreviated Sensor (SENSOR_BRIEF) ACF message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The SENSOR_BRIEF message has an **acf_msg_type** field of ACF_SENSOR_BRIEF (see Table 22).

The SENSOR_BRIEF message defines the following message-specific fields:

— **r** (reserved): 1 bit

— **num_sensors**: 1 octet

— **sz** (sensor value size): 2 bits

— **sensor_group**: 6 bits

— **sensor_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 66.

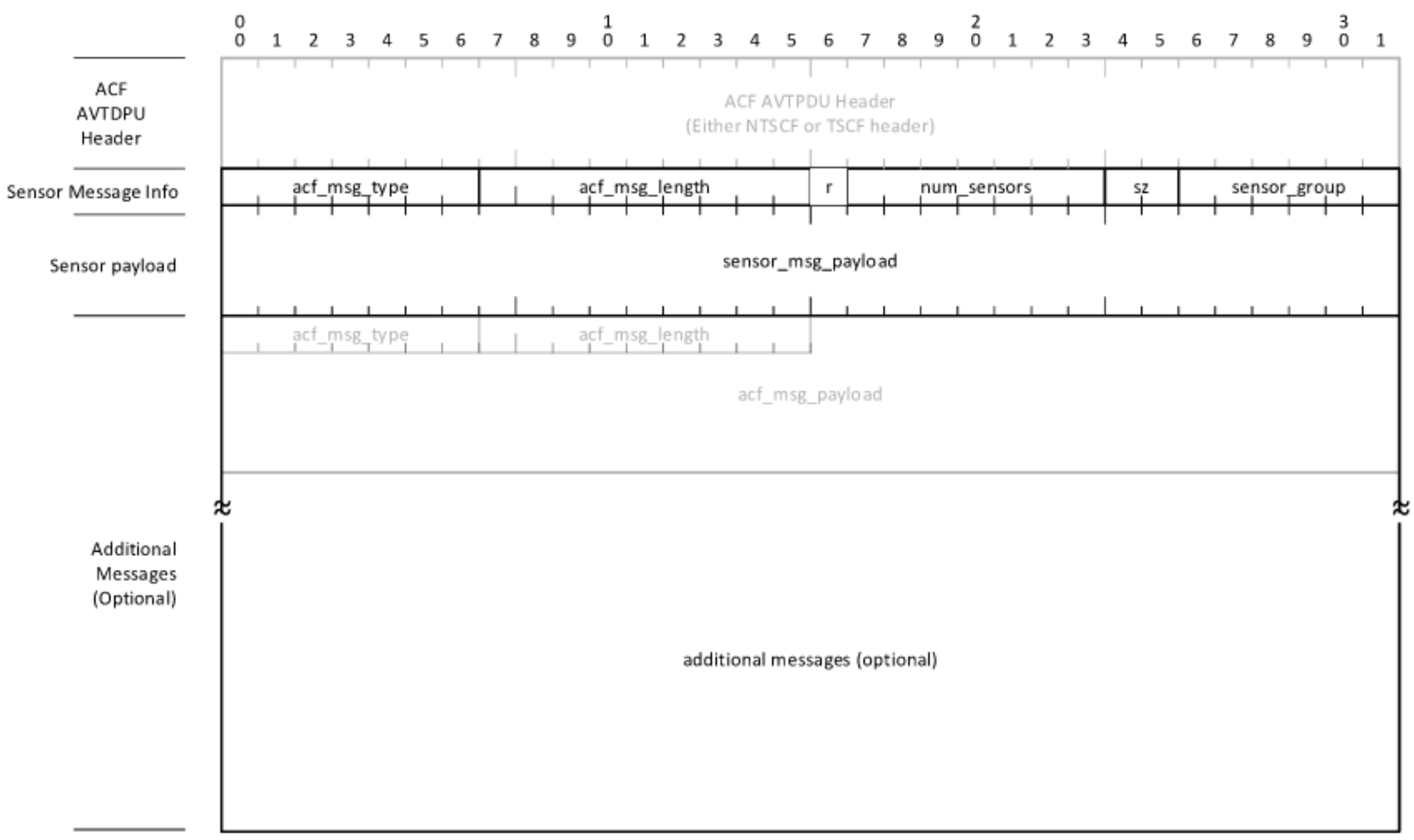


Figure 66—Abbreviated Sensor (SENSOR_BRIEF) message

The SENSOR_BRIEF message has the same fields as the standard SENSOR message (see 9.4.10), except that it does not have the 2-quadlet timestamp field. The SENSOR_BRIEF message is designed to minimize

Copyright © 2016 IEEE. All rights reserved.

PDU size and maximize network efficiency when carrying sensor data for which the acquisition timestamp is not necessary.

9.4.12 AECP message

9.4.12.1 Overview

The AVDECC Enumeration and Control Protocol (AECP) ACF message type is used to transport an IEEE 1722.1 AECPDU as defined in subclause 9.2.1 of IEEE Std 1722.1-2013. The AECP ACF message is useful in control applications using an IEEE 1722.1 controller. Use cases include control of audio and staging devices.

The AECP message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits

— **acf_msg_length**: 9 bits

The AECP message has an **acf_msg_type** field of ACF_AECP (see Table 22).

The AECP message defines the following message-specific fields:

— **reserved:** 2 octets

— **aecp_msg_payload**: 0 to *n* quadlets

These fields are shown in Figure 67.

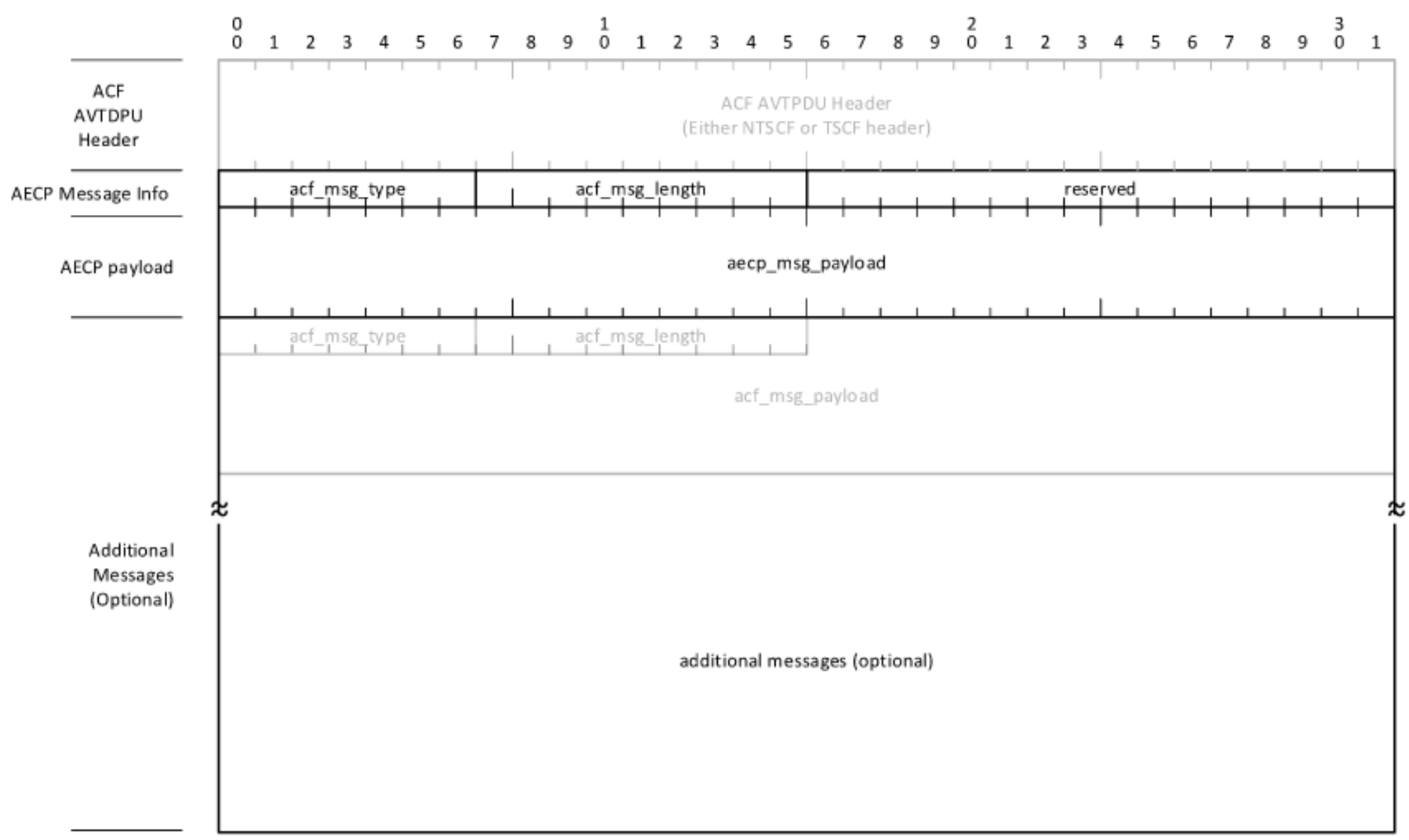


Figure 67—AECP message

Copyright © 2016 IEEE. All rights reserved.

9.4.12.2 aecp_msg_payload field

The **aecp_msg_payload** field contains the entire AECPDU message, as defined in subclause 9.2.1 of IEEE Std 1722.1-2013.

9.4.13 Ancillary Data message

9.4.13.1 Overview

The Ancillary Data ACF message provides transport of SMPTE ST 291-1 [B27] ancillary data over a TSN network. It is useful in digital video systems. Use cases include interconnection of the ancillary data feeds of two digital video systems.

Ancillary data provides supporting information for the main video data of a digital video stream, including transport of audio and closed captioning data. As defined in SMPTE ST 291-1, it rides in unused space of the main video data stream. Ancillary data is broken into "streams" and these streams are then broken into ancillary data packets. Ancillary data streams can be of arbitrary length, while ancillary data packets have a limited length. Each ancillary data packet has a header of 6 octets and a payload containing a maximum of 255 octets.

Ancillary data may be contained in any area defined as available for ancillary data, including individual video scan lines. An ancillary data message may be contained entirely within a single video line, or it may span multiple video lines. The meaning and interpretation of the Ancillary data is defined in separate application documents.

9.4.13.2 Ancillary Data encapsulation

9.4.13.3 Overview

The Ancillary Data message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits
— **acf_msg_length**: 9 bits

The Ancillary Data message has an **acf_msg_type** field of ACF_ANCILLARY (see Table 22).

In addition, if the Ancillary Data ACF's AVTPDU uses a TSCF header (see 9.3), it is recommended that

— All ACF messages contained in the AVTPDU refer to the same video line, and
— The timestamp refer to the beginning of the video line to which the ACF messages in the AVTPDU are related.

The Ancillary Data message defines the following message-specific fields:

— **pad** (padding length): 2 bits
— **reserved**: 10 bits
— **mode**: 2 bits
— **fp** (first packet): 1 bit

Copyright © 2016 IEEE. All rights reserved.

— **lp** (last packet): 1 bit

— **line_number**: 2 octets

— **DID**: 1 octet

— **SDID/DBN**: 1 octet

— **ancillary_data_payload**: 0 to *n* octets

These fields are shown in Figure 68.

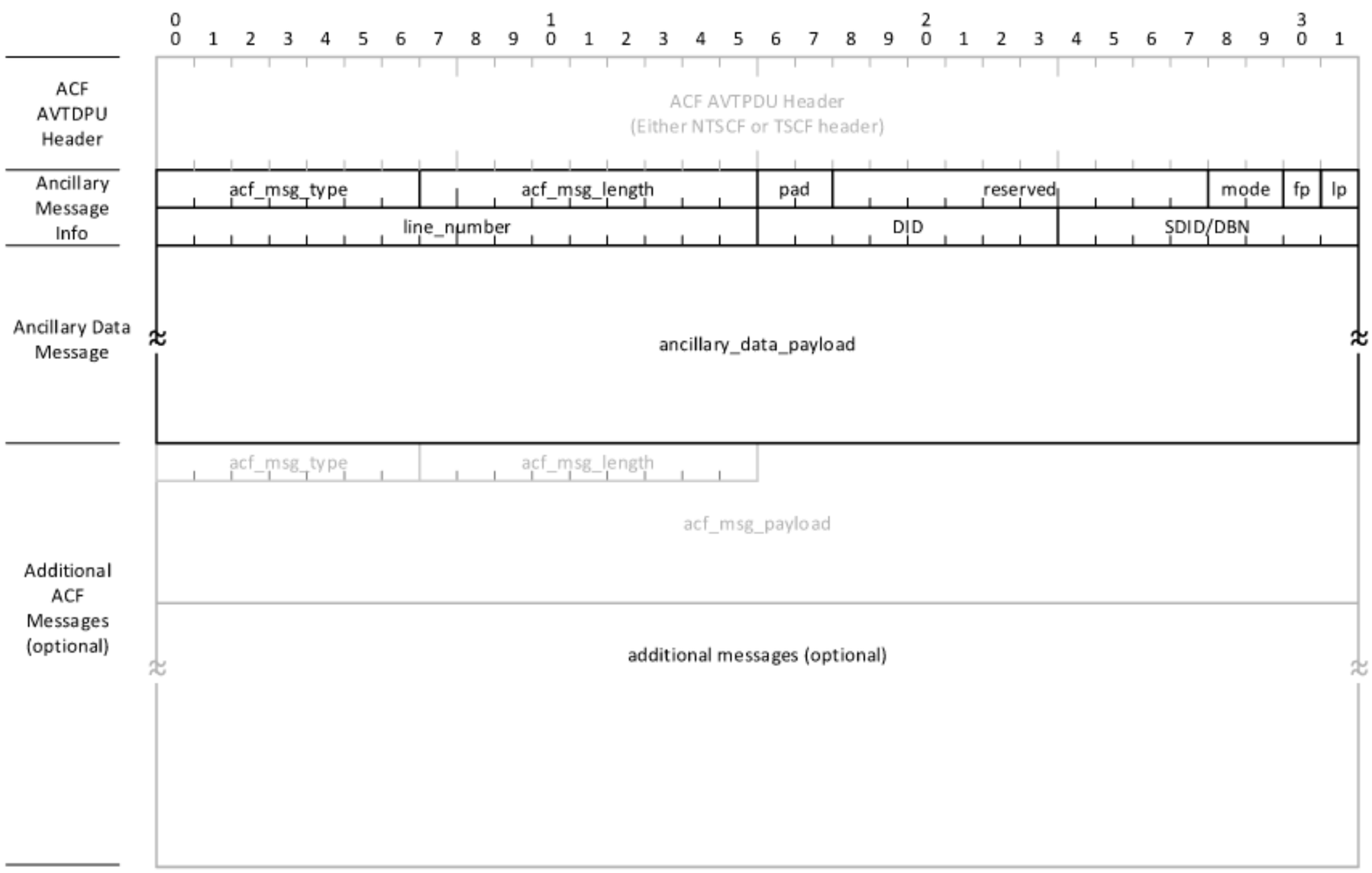


Figure 68—Ancillary Data PDU Format

9.4.13.3.1 pad (padding length) field

The **pad** field indicates the length of padding at the end of the message payload in octets. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload.

9.4.13.3.2 mode field

The **mode** field specifies the format of the ancillary data contained in the ancillary_data_payload field, as defined by Table 25. See 9.4.13.3.8 for further details on the structure of payload data.

Copyright © 2016 IEEE. All rights reserved.

Table 25—Ancillary data modes

Value	Name	Description
0_{16}	ANC_8BIT	8-bit mode
1_{16}	ANC_10BIT	10-bit mode
2 to 3_{16}	—	Reserved

9.4.13.3.3 fp (first packet) field

The **fp** field is set to one (1) for the first ancillary data packet of an ancillary data stream. For all other ancillary data packets of an ancillary data stream, the **fp** field is set to zero (0).

If an ancillary data stream contains only one ancillary data packet, then the **fp** field is set to one (1).

9.4.13.3.4 lp (last packet) field

The **lp** field is set to one (1) for the last ancillary data packet of an ancillary data stream. For all other ancillary data packets of an ancillary data stream, the **lp** field is set to zero (0).

If an ancillary data stream contains only one ancillary data packet, then the **lp** field is set to one (1).

NOTE—The **fp** and **lp** fields encompass all information contained in the ancillary data flag word (as defined in SMPTE ST 291-1:2011 [B27]), and as such, no additional field is needed in the ancillary data ACF message.

9.4.13.3.5 line_number field

The **line_number** field contains the line number carrying this ancillary data packet in the associated video stream. The **line_number** assumes a progressive video structure, not interlaced.

9.4.13.3.6 DID field

The **DID** field contains the contents of the DID field of the original ancillary data packet as defined in SMPTE ST 291-1:2011 [B27].

9.4.13.3.7 SDID/DBN field

The **SDID/DBN** field contains the contents of the DBN field of a Type 1 ancillary data packet, or the SDID field of a Type 2 ancillary data packet as defined in SMPTE ST 291-1:2011 [B27].

9.4.13.3.8 ancillary_data_payload field

9.4.13.3.8.1 Overview

The **ancillary_data_payload** field contains the ancillary data packet user data.

As described in SMPTE ST 291-1:2011 [B27] ancillary data packet payload has a maximum size of 255 octets. A Talker shall encapsulate each ancillary data packet into the **ancillary_data_payload** field of a single ACF message. Multiple ancillary data packets, in the form of ACF messages, may be placed into a

Copyright © 2016 IEEE. All rights reserved.

single ACF AVTPDU as long as the size of the ACF AVTPDU does not exceed the MTU. For large ancillary data streams, multiple ACF AVTPDUs may be needed to transport the entire stream. When a complete ancillary data packet does not fit in the remaining space of an ACF AVTPDU, the complete ancillary data packet should be encapsulated in a new ACF AVTPDU; Ancillary data packets shall not be split across multiple ACF messages.

The user data is either in 8-bit or 10-bit words. The following subclauses describe how to encapsulate the user data words for each case.

9.4.13.3.8.2 8-bit packet encoding

In an 8-bit packet encoding, the low-order 8-bits from each octet in the ancillary data packet payload are placed in each octet of the **ancillary_data_payload** field. The bits in the ancillary data payload representing parity, inverted parity, and the complete check sum from the ancillary data packet are not included.

User data words are mapped directly into the GPC message in the order they appear in the ancillary data packet.

Messages using 8-bit packet encoding have the **mode** field set to ANC_8BIT.

An ancillary data message Listener that is regenerating ancillary data streams shall generate the required ancillary data packet flag words, the checksum word, parity and inverted parity bits for *all* ancillary data packet words in compliance with SMPTE ST 291-1:2011 [B27]. The Listener can generate the data count (DC) word in the ancillary data packet header from the length of the **ancillary_data_payload** field.

9.4.13.3.8.3 10-bit coding

The 10-bit coding mode is designed to allow "pass through" of ancillary data packet data without changing the initial parity or checksum. Mapping of 10-bit words into quadlets of the ancillary data ACF message shall be performed according to Figure 69. Three 10-bit words are packed into a 32-bit word, leaving the two (2) least-significant bits unused. A Talker shall set these 2 bits to zero (0). In 10-bit mode, the Talker shall include the 10-bit checksum word as the final 10-bit ANC data block in the payload.

Messages of this type have the **mode** field set to ANC_10BIT.

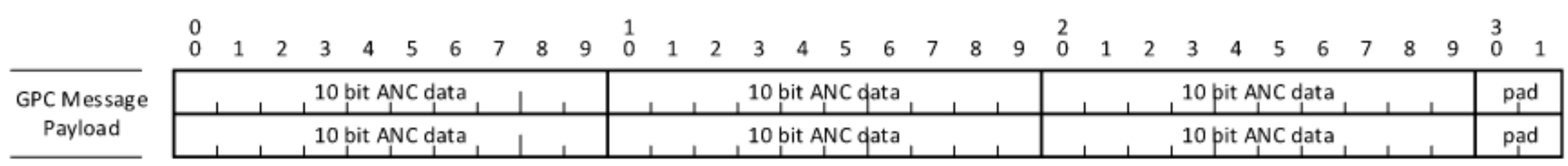


Figure 69—Example of 10-bit ANC mapping in GCF

An ancillary data message Listener that is regenerating ancillary data streams shall generate the required ancillary data packet flag words, parity flags for the DID and SDID according to SMPTE ST 291-1:2011 [B27]. The Talker shall pass on all 10 bits of information (parity, inverted parity) for all other words, directly from the **ancillary_data_payload** field, including the original checksum. The Listener can generate the data count (DC) word in the ancillary data packet header from the length of the **ancillary_data_payload** field.

Copyright © 2016 IEEE. All rights reserved.

10. Clock Reference Format

10.1 Overview

The Clock Reference Format (CRF) allows for the dissemination of event timing information to multiple TSN devices. CRF supports formats for audio media clock, video line, video frame, machine cycles, and other user-defined event timing. Although not required for TSN system operation, CRF is a feature that can simplify and ease overall system design by minimizing complexity.

CRF has several optimizations that make it more efficient than using a standard AVTP audio or video stream to distribute timing information. These include:

— CRF uses a compact format due to only including timing information

— CRF is tolerant of lost packets since clocks free-wheel between defined clock points

— CRF data can be sent at a variety of rates depending on the use case

— Multiple CRF streams can be used to create multiple media clock domains on a single network

— CRF can be expanded using the CRF_USER type to accommodate use cases outside those described by this format

10.2 Background

The distribution of a common clock in an audio/video/control system is used to synchronize events within a system. In audio systems it is desirable to synchronize the media clock throughout the system. In a video-distribution system it is desirable to synchronize frames and lines throughout the system. Likewise, in an automated control system it is desirable to synchronize machine cycle times.

The distributed common clock can be used by AVTP Talkers within a system to synchronize stream data such that an AVTP Listener can receive streams from multiple Talkers and know that the streams are all synchronized with each other. This allows use cases such as video switching, audio mixing, and synchronized machine automation to be designed with less complexity. In a video system, the switching from one camera to another can be simplified if both cameras are genlocked. Likewise, sensors from an automated control system can sample multiple sensors at the same time if their machine cycles are synchronized.

10.3 Audio example

The synchronization of media clocks within multiple audio Talkers allows a single audio Listener to receive audio from these multiple Talkers without the use of sample rate conversion. An example of this use case can be shown with the use of an audio-mixing console. A mixing console is expected to receive audio signals from a wide variety of sources and combine or mix these signals into a standard format such as stereo or 5.1 audio. The job of the mixing console is simplified if it is not required to accept a wide variety of media clock sources and sample rate convert each of these sources to be aligned with the mixing console's internal media clock.

This example is illustrated in Figure 70.

Copyright © 2016 IEEE. All rights reserved.

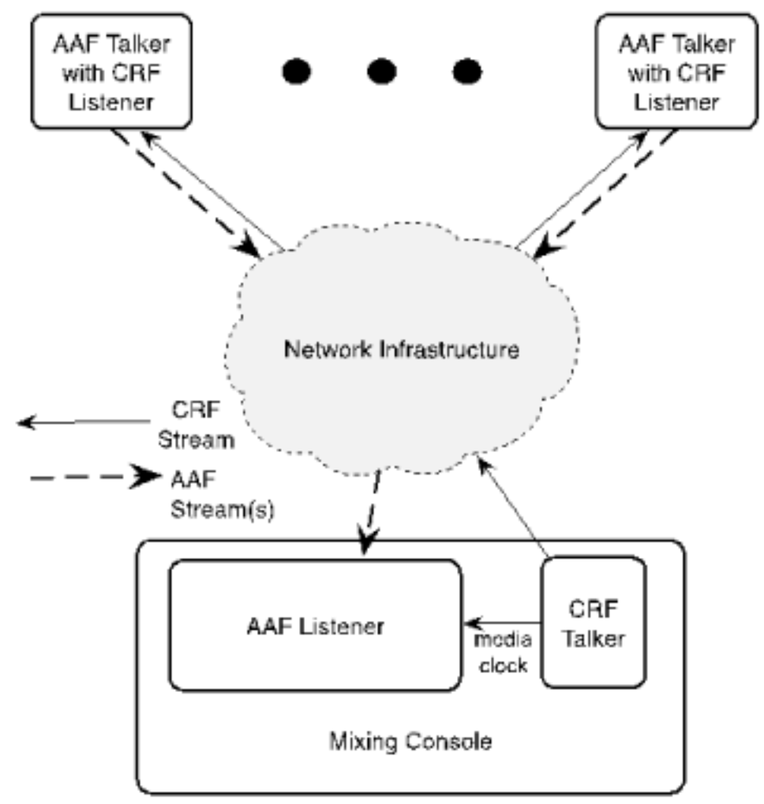


Figure 70—CRF audio mixer example

10.3.1 Clock reference functional blocks

Figure 71 shows a simplified example of the various functional blocks that exist in CRF-capable devices.

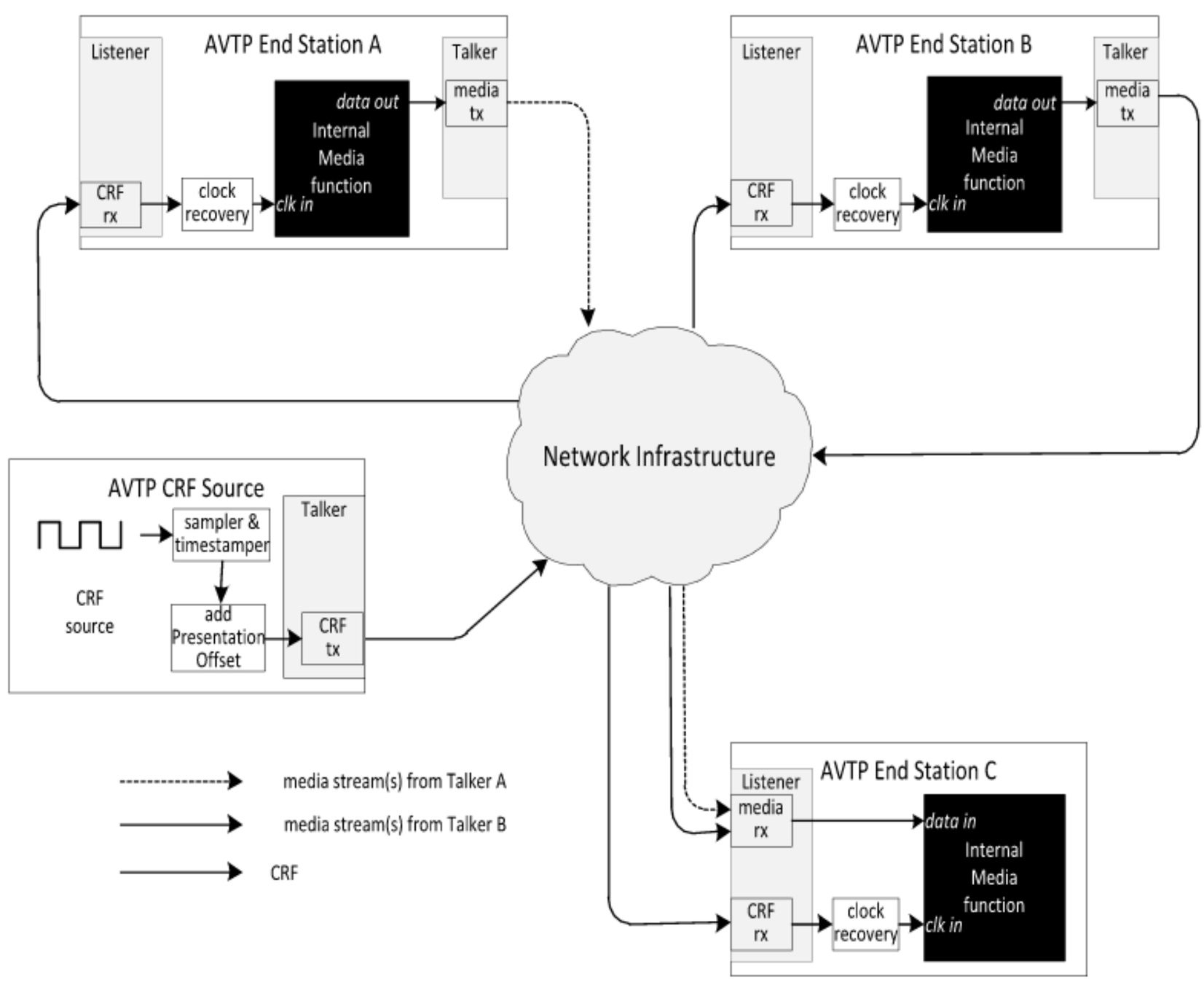


Figure 71—Time-Sensitive Network with CRF stream

Copyright © 2016 IEEE. All rights reserved.

In Figure 71, End Station C may recover media clock information from Stream A, Stream B, or the CRF stream. Using the CRF stream is often a good choice because Stream A or Stream B may or may not be running at any given time, while the CRF stream could be made to always be available since CRF streams are low bandwidth and don't require a stream reservation.

10.4 Clock Reference Format Data encapsulation

10.4.1 Overview

CRF streams consist of an AVTPDU alternative header, followed by a type specific header, then one or more timestamps.

CRF streams are not required to be reserved.

CRF AVTPDUs shall use the alternative header and shall have a **subtype** field of CRF (see 4.4.3.2).

This encapsulation uses the following common alternative fields (see 4.4.3 and 4.4.6 for definitions):

— **version**: 3 bits

This encapsulation uses the **h** and **alternative_data_payload** fields of the alternative header to define the following format-specific fields:

— **sv** (stream_id valid): 1 bit

— **mr** (media clock reset): 1 bit

— **r** (reserved): 1 bit

— **fs** (frame sync): 1 bit

— **tu** (timestamp uncertain): 1 bit

— **sequence_num**: 1 octet

— **type**: 1 octet

— **stream_id**: 8 octets

— **pull**: 3 bits

— **base_frequency**: 29 bits

— **crf_data_length**: 2 octets

— **timestamp_interval**: 2 octets

— **crf_data**: 0 to *n* octets

These fields are shown in Figure 72.

Copyright © 2016 IEEE. All rights reserved.

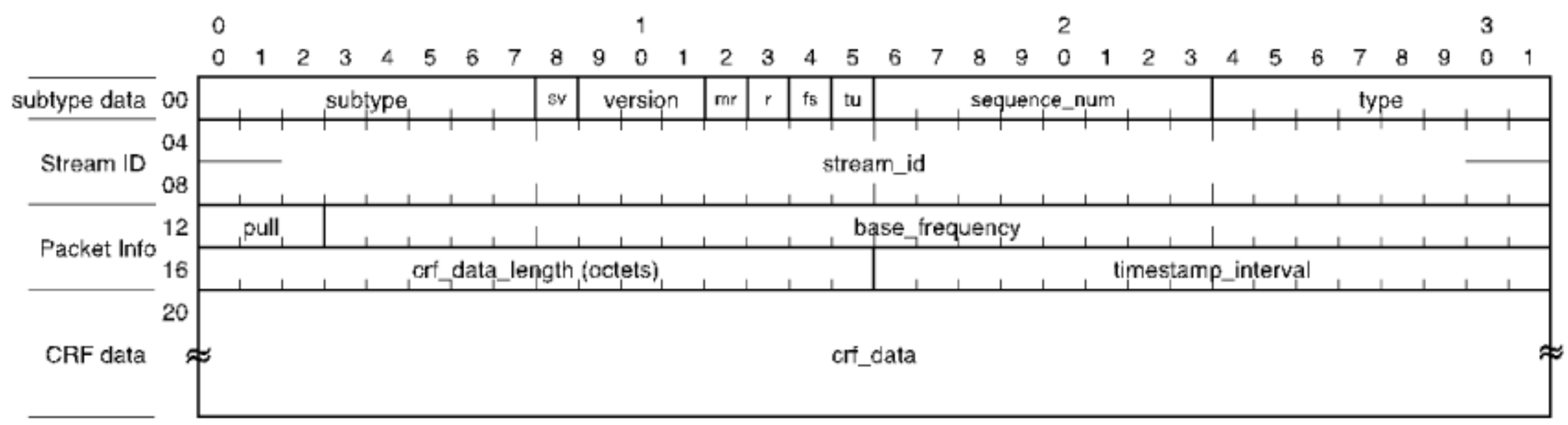


Figure 72—Clock Reference PDU Format

10.4.2 sv (stream_id valid) field

The **sv** (stream_id valid) field is used to indicate whether the 8-octet **stream_id** field contains a valid StreamID.

10.4.3 mr (media clock restart) field

The **mr** (media clock restart) field is a 1-bit field used by a CRF Talker to indicate that there has been a change in the source of the media clock that is providing the samples used in the CRF stream. An example of this would be a change from one audio/video/control input to a different audio/video/control input where the media clock is supplied from the input (i.e., switching from one Sony/Philips Digital Interconnect Format [S/PDIF] input to a different S/PDIF input).

This bit shall be used by the CRF Listener to:

— Adjust its own recovered media clock quickly (10.6) to the new media clock represented in the CRF stream. This kind of change need not be seamless but allows the Listener to take any appropriate action to minimize the disruption to the stream.

— Toggle the **mr** bit in any outgoing media streams that are deriving timestamps from the CRF stream.

This bit is toggled by the Talker each time a media clock restart is needed, and it stays at its new value until a new media clock restart is needed. This toggle helps reduce the likelihood that the media clock restart is missed if it were to appear in a single CRF AVTPDU that just happens to be dropped. Once this bit toggles, it shall remain in its new state for a minimum of eight (8) CRF AVTPDUs for a given continuous stream.

If a Listener is receiving both a media stream and a CRF stream, only the **mr** bit from the stream being used by the Listener for recovering the media clock is valid.

10.4.4 fs (frame sync) field

The **fs** field enables a clock to also convey a frame-synchonized clock. Usage of this field is defined per type under the **crf_data** field (10.4.13).

10.4.5 tu (timing uncertain) field

The **tu** field behaves the same way as the **tu** field from the common stream header (4.4.4.7) except that it applies to the timestamps contained in the **crf_data** field.

Copyright © 2016 IEEE. All rights reserved.

10.4.6 sequence_num field

The sequence number field (**sequence_num**) is a 1-octet field used to indicate the sequence of CRF AVTPDUs in a stream by the Talker. This field may start at any value for the first CRF AVTPDU of a stream and shall be incremented by one (1), with wrapping, on each subsequent CRF AVTPDU of a stream. This field is to wrap from 11111111_2 to 00000000_2 (FF_{16} to 00_{16}).

The value of this field is determined by the Talker. Listeners can use the sequence number to detect CRF AVTPDUs lost in transit.

A Listener may join a stream at any time after the stream has started, and therefore, a Listener shall be able to tolerate a starting **sequence_num** of any value.

10.4.7 type field

The **type** field indicates the type of timestamp represented by the CRF AVTPDU and the format of the CRF payload. Defined values for this field are given in Table 26.

Table 26—CRF types

Value	Name	Description	Clause
0_{16}	CRF_USER	User specified	10.4.13.2
1_{16}	CRF_AUDIO_SAMPLE	Audio sample timestamp	10.4.13.3
2_{16}	CRF_VIDEO_FRAME	Video frame sync timestamp	10.4.13.4
3_{16}	CRF_VIDEO_LINE	Video line sync timestamp	10.4.13.5
4_{16}	CRF_MACHINE_CYCLE	Machine cycle timestamp	10.4.13.6
5 to FF_{16}	Reserved	Reserved	—

10.4.8 stream_id field

The **stream_id** field contains the 8-octet StreamID associated with the CRF AVTPDU. The **stream_id** field is used for stream identification.

All CRF AVTPDUs shall contain a valid StreamID in the **stream_id** field, as defined in subclause 35.2.2.8.2 of IEEE Std 802.1Q-2014, and as shown for convenience in Figure 73.

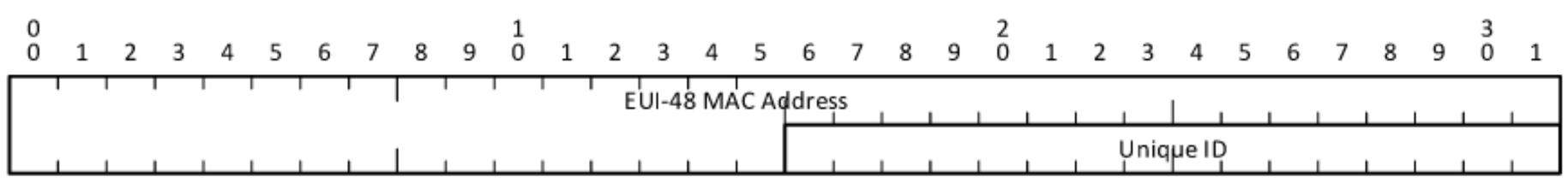


Figure 73—CRF StreamID Format

10.4.9 pull field

The **pull** field specifies the multiplier modifier of the **base_frequency** field that is required to calculate the appropriate nominal sampling rate of the CRF stream.

The **pull** field may have one of the values defined in Table 27.

Copyright © 2016 IEEE. All rights reserved.

Table 27 —pull field values

Value	Description
0_{16}	Multiply **base_frequency** field by 1.0
1_{16}	Multiply **base_frequency** field by 1/1.001
2_{16}	Multiply **base_frequency** field by 1.001
3_{16}	Multiply **base_frequency** field by 24/25
4_{16}	Multiply **base_frequency** field by 25/24
5_{16}	Multiply **base_frequency** field by 1/8
6 to 7_{16}	Reserved

10.4.10 base_frequency field

The **base_frequency** field defines the nominal base sampling rate in Hertz (Hz). The base sampling rate shall be from 1 Hz to 536 870 911 ($1FFFFFFF_{16}$) Hz. The value of this field is augmented by the **pull** field value.

10.4.11 crf_data_length field

The 2-octet **crf_data_length** field contains the length (in octets) of the **crf_data** field. Because all timestamps are 8 octets, the **crf_data_length** field shall be a value that is a multiple of 8 (e.g., 0, 8, 16, etc.).

10.4.12 timestamp_interval field

The **timestamp_interval** field is a 2-octet field that indicates the number of events between each timestamp in the **crf_data** field where an event is defined by the **type** of the clock. The value of this field shall be static for the duration of the life of the stream and shall be nonzero.

10.4.13 crf_data field

10.4.13.1 Overview

The **crf_data** field contains the main content of the CRF stream. The format of this stream is dependent on the value of the **type** field.

A CRF timestamp is conceptually the same as an **avtp_timestamp**, which is defined in 4.4.4.9. However, in order to span a longer time scale than an **avtp_timestamp**, a CRF timestamp contains an 8-octet timestamp in units of nanoseconds. A CRF timestamp derived from gPTP time has a maximum value of $2^{64} - 1$ ns, which rolls over approximately every 584 years. Since CRF timestamps are based on gPTP time, which may not be related to any actual time, it is possible for a CRF stream to encounter a rollover event even though the life of the stream is less than 584 years.

The CRF timestamp is constructed from the gPTP **seconds** and **nanoseconds** fields by the formula given in Equation (13):

$$\text{message_timestamp} = (\text{AS_sec} \times 10^9 + \text{AS_ns}) \bmod 2^{64} \tag{13}$$

Copyright © 2016 IEEE. All rights reserved.

where

AS_sec the gPTP **seconds** field

AS_ns the gPTP **nanoseconds** field

For CRF AVTPDUs with a **type** value of CRF_AUDIO_SAMPLE, CRF_VIDEO_FRAME, CRF_VIDEO_LINE, or CRF_MACHINE_CYCLE the **crf_data** field is structured as shown in Figure 74.

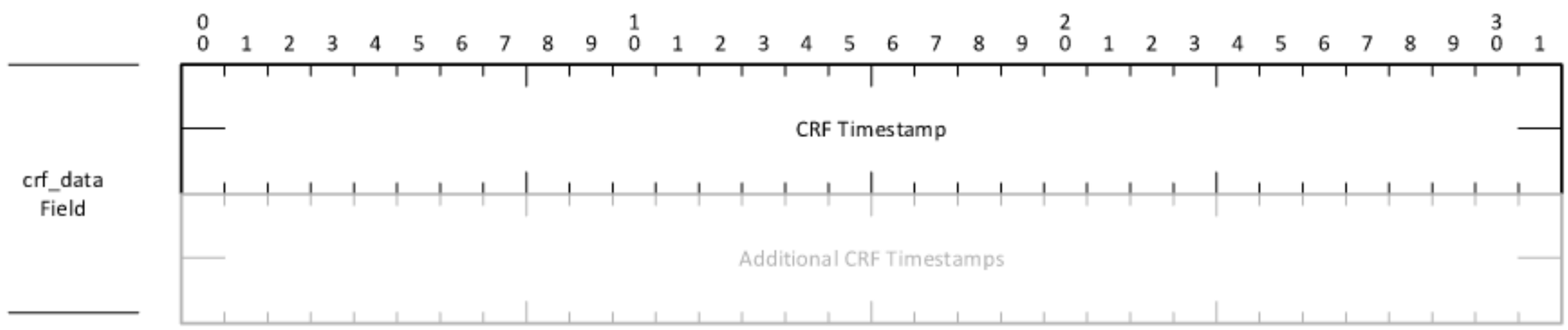


Figure 74—Payload of crf_data for standard types

10.4.13.2 User-specified type

CRF AVTPDUs with a **type** field of CRF_USER (Table 26) are used when a higher level application or protocol has defined a clocking type that is not included in the CRF types defined in Table 26 besides CRF_USER. The contents of the **crf_data** field may contain additional header information and contain one or more CRF timestamps as defined in 10.6.

10.4.13.3 Audio sample type

CRF AVTPDUs with a **type** field of CRF_AUDIO_SAMPLE (Table 26) convey the sampling time of an audio sample clock. The **crf_data** field contains one or more CRF timestamps as defined in 10.6.

The audio sample type does not contain a frame-synchronized clock and the **fs** field shall be set to zero (0).

10.4.13.4 Video frame sync type

CRF AVTPDUs with a **type** field of CRF_VIDEO_FRAME (Table 26) convey the start of frame time of a video signal. The **crf_data** field contains one or more CRF timestamps as defined in 10.6.

The video frame sync type does not contain a frame-synchronized clock and the **fs** field shall be set to zero (0).

10.4.13.5 Video line sync type

CRF AVTPDUs with a **type** field of CRF_VIDEO_LINE (Table 26) convey the start of line time of a video signal. The **crf_data** field contains one or more CRF timestamps as defined in 10.6.

The video line sync type carries a frame-synchronized clock. The **fs** field is used to indicate when the first timestamp is coincident with the start of a video frame. When both horizontal and vertical timing are being conveyed in a single CRF stream, the **fs** field shall be set to one (1) when the first timestamp in the **crf_data** field is also the start of a video frame and zero (0) otherwise. If horizontal and vertical timing are

Copyright © 2016 IEEE. All rights reserved.

being conveyed in separate CRF streams, the **fs** field may be set to one (1) when the first timestamp in the **crf_data** field is also the start of a video frame and zero (0) otherwise.

NOTE—Due to the relationship between the number of lines in a frame, the CRF **timestamp_interval**, and timestamps per CRF AVTPDU, it is possible that the **fs** field will not indicate the start of a video frame. The specific CRF timestamp indicating the start of video frame may be skipped due to a **timestamp_interval** greater than one (1), or it may not be the first CRF timestamp in an AVTPDU. It is possible to design a CRF_VIDEO_LINE CRF stream that does reliably indicate the start of video frame using the **fs** field. It is also possible to use two CRF streams, a CRF_VIDEO_LINE and CRF_VIDEO_FRAME stream, to reliably specify both video line and frame information.

10.4.13.6 Machine cycle type

CRF AVTPDUs with a **type** field of CRF_MACHINE_CYCLE (Table 26) convey the start of cycle time of a machine cycle. The **crf_data** field contains one or more CRF timestamps as defined in 10.6.

The machine cycle type does not contain a frame-synchronized clock and the **fs** field shall be set to zero (0).

10.5 CRF transmission rates

CRF AVTPDUs can be transmitted at various rates based on the requirements of specific installations or vendors. For interoperability reasons, suggested rate requirements for a Listener are given in Table 28, Table 29, Table 30, and Table 31, based on the CRF types in Table 26. Rates not included in these tables are permitted.

If an end station supports the CRF_AUDIO_SAMPLE type, some recommended CRF rates are shown in Table 28. A 300-Hz rate is included in the recommended rates since 300 Hz is the greatest common divisor that can be used to recover all common audio sample rates.

Table 28—Recommended CRF transmission rates for CRF_AUDIO_SAMPLE type

Rate (Hz)	Timestamps per CRF AVTPDU	CRF AVTPDUs per second	Timestamps per second	Timestamp interval
300	6	50	300	1
44 100	6	50	300	147
48 000	6	50	300	160
88 200	6	50	300	294
96 000	6	50	300	320
176 400	6	50	300	588
192 000	6	50	300	640

Table 29 shows some recommended rates for use with the CRF_VIDEO_FRAME type.

Copyright © 2016 IEEE. All rights reserved.

Table 29—Recommended CRF transmission rates for CRF_VIDEO_FRAME type

Format	Frame rate (Hz)	Timestamps per second	Timestamps per CRF AVTPDU	CRF AVTPDUs per second	Video frames per CRF AVTPDU	Timestamp interval
SD625	25.00	25	1	25	1	1
SD525	29.97	29.97	1	29.97	1	1
1080i50	25.00	25	1	25	1	1
1080p50	50.00	50	2	25	2	1
1080p100	100.00	100	4	25	4	1
1080i59.94	59.94	59.94	2	29.97	2	1
1080i120	120.00	120	4	30	4	1

Table 30 shows some recommended rates for use with the CRF_VIDEO_LINE type when using the SVF.

Table 30—Recommended CRF transmission rates for CRF_VIDEO_LINE using SVF

Format	Frame rate (Hz)	Lines	Timestamps per second	Timestamps per CRF AVTPDU	CRF AVTPDUs per second	CRF AVTPDUs per video frame	Timestamp interval
525i	29.97	525	3146.85	5	629.37	21	5
625i	25.00	625	3125	5	625	25	5
1080i	59.94	1125	13 486.5	5	2697.3	45	5
1080i	50.00	1125	11 250	5	2250	45	5
720p	59.94	750	8991	5	1798.2	30	5
720p	50.00	750	7500	5	1500	30	5
1080p	23.98	1125	5395.5	5	1079.1	45	5
1080p	59.94	1125	13 486.5	5	2697.3	45	5
1080p	50.00	1125	11 250	5	2250	45	5

Table 31 shows some recommended rates for use with the CRF_VIDEO_LINE type when using RVF.

Table 31—Recommended CRF transmission rates for CRF_VIDEO_LINE using RVF

Format	Frame rate (Hz)	Lines	Timestamps per second	Timestamps per CRF AVTPDU	CRF AVTPDUs per second	CRF AVTPDUs per video frame	Timestamp interval
525i	29.97	480	3596.4	2	1798.2	60	4
625i	25.00	576	3600	2	1800	72	4
1080i	59.94	1080	16 183.8	2	8091.9	135	4
1080i	50.00	1080	13500	2	6750	135	4
720p	59.94	720	10 789.2	2	5394.6	90	4
720p	50.00	720	9000	2	4500	90	4
1080p	23.98	1080	6474.6	2	3237.3	135	4
1080p	59.94	1080	16 183.8	2	8091.9	135	4
1080p	50.00	1080	13 500	2	6750	135	4

10.6 Interaction between CRF timestamps and Media Stream timestamps

CRF timestamps convey the timing of media events (e.g., audio sample, video lines, video frames, machine cycles), allowing all Listeners of a given CRF stream to reproduce the same media events. CRF timestamps convey timing points that are offset from the original source clock by a nominal, positive delta, such that when they arrive in the Listener they can be simply compared to the local gPTP clock at the Listener.

Copyright © 2016 IEEE. All rights reserved.

If a CRF stream is not sent as a reserved stream then network latency can be unpredictable. In the case of an unreserved CRF stream, it may be that CRF timestamps arrive that are in the past compared with the current gPTP time; however, it is still possible to use these CRF timestamps to recreate a media clock.

A CRF Talker may supply multiple timestamps within a single CRF AVTPDU to allow the Listener to lock its media clock quickly to the supplied clock by processing all of the timestamps within the CRF AVTPDU. Once the Listener has locked to the remote clock, it may continue to process all of the timestamps received within the CRF AVTPDUs or it may process only a subset of the timestamps received as necessary to maintain lock while reducing the processing overhead.

If CRF timestamps are lost due to network packet loss, the media clock free-wheels until the CRF stream resumes and new CRF timestamps are available.

10.7 Requirements of CRF timestamp values

The first CRF timestamp contained in the **crf_data** field of a CRF AVTPDU shall be set to a value that is offset, in positive nanoseconds, from the transmit time plane (see Figure 6) at the Talker by the Max Transit Time of the network, rounded up to the nearest multiple of the media clock as shown in Equation (14). This method places the timestamp on a multiple of the media clock, rounding up so that the Max Transit Time is not violated. Refer to 4.3.2 for more information on Max Transit Time and how it is determined.

$$T_{\mathrm{CRF}} = T_{\mathrm{S}} + \left(\left\lceil \frac{TT_{\max}}{p} \right\rceil \times p \right) + T_C \tag{14}$$

where

T_{CRF} the CRF timestamp that is placed in the CRF AVTPDU

T_{s} the original timestamp, sampled at the source

$TT_{\max}$ the Max Transit Time value chosen for the network

P the nominal period of the clock source

T_C the amount of time that samples spend accumulating in the Talker's transmit buffer

$\lceil x \rceil$ the bracket-like symbols represent the *ceiling* function

When a CRF AVTPDU contains multiple CRF timestamps in its payload the values of the timestamps shall increase monotonically from the first timestamp to the last timestamp in the CRF AVTPDU.

NOTE 1—A rollover of the timestamp field is still considered to be increasing monotonically.

NOTE 2—If the **tu** field is set to one (1), there may be discontinuities in the CRF timestamps. In this case, the media clock will free-wheel until the CRF timestamps and the gPTP clock stabilize.

A CRF Talker that also has internal media functionality beyond producing the CRF stream derives its media clock from the CRF timestamp values that it outputs to other end stations in the system. This will allow for the Talker to be clocked in the same manner as other devices on the network using the same CRF stream. Essentially, these devices are considered multi-function devices, where each function must be considered fully independent such that each function could be done elsewhere in the network and identical results will occur. This is shown in Figure 75.

Copyright © 2016 IEEE. All rights reserved.

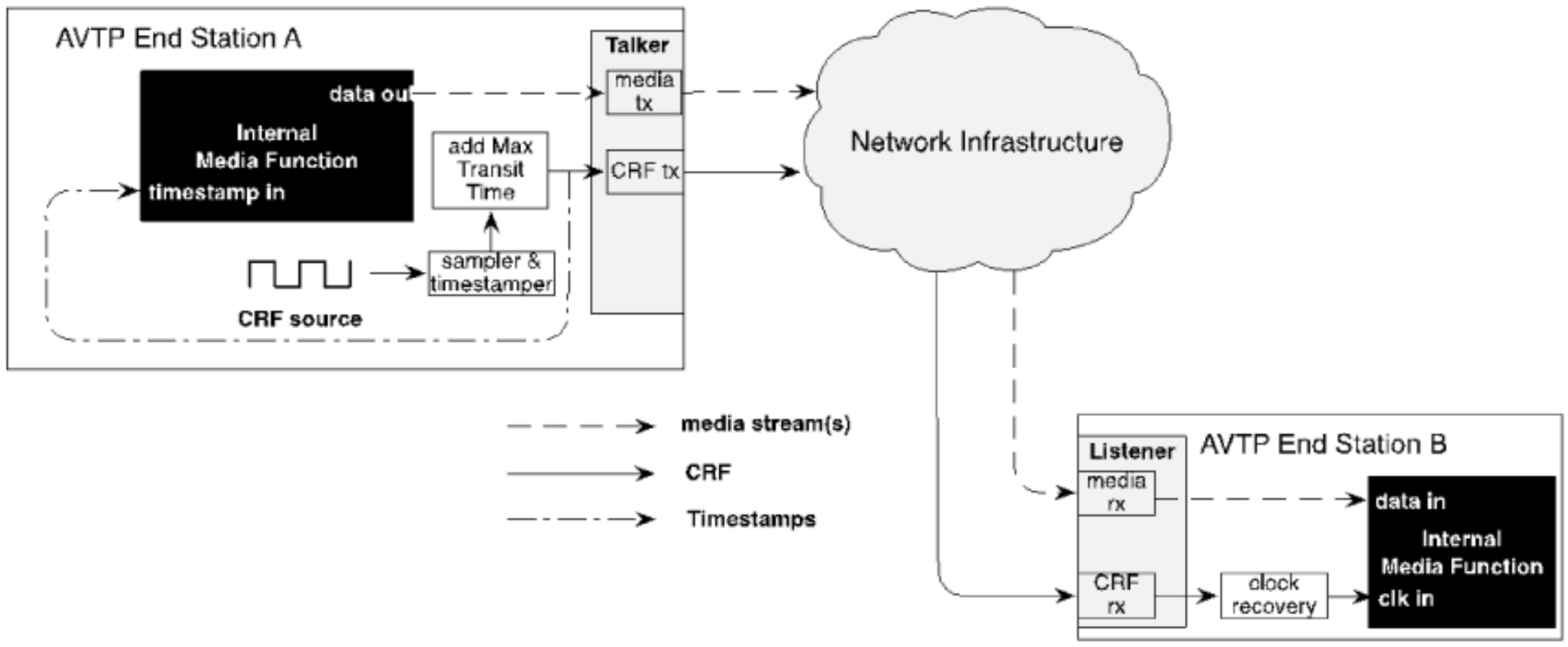


Figure 75—CRF Talker using its own CRF data

10.8 Relationship between CRF timestamps and stream timestamps

All Talkers that recover media clocks from a CRF stream use **avtp_timestamp** values that align with the common clock established by the CRF stream. The maximum delta between timing points from the received CRF stream and **avtp_timestamp** values sent by Talker media streams shall not exceed +/− 5.0% of the media sample period, as shown by the equation in Equation (15).

$$n \times P_s - \frac{P_s}{20} < T_{offset} < n \times P_s + \frac{P_s}{20} \tag{15}$$

where

T_{offset} the timestamp offset in nanoseconds between the AVTP Presentation Timestamp of the media stream and the timestamp of the CRF stream

n a positive integer chosen for the implementation

P_s the sample period of the CRF stream in nanoseconds

NOTE—The 5% tolerance for alignment between output media phase and CRF phase is consistent with AES11-2009 [B5].

All Listeners slaving to a CRF stream accept streams (within the same CRF domain) whose **avtp_timestamp** values align to the common clock within +/− 25% of a sample period or less, as shown by the Equation (16). If the incoming **avtp_timestamp** values are misaligned to the common clock by more than +/− 25% of a sample period, the Listener may interpret the stream as invalid.

$$n \times P_s - \frac{P_s}{4} < T_{offset} < n \times P_s + \frac{P_s}{4} \tag{16}$$

where

T_{offset} the timestamp offset in nanoseconds between the AVTP Presentation Timestamp of the media stream and the timestamp of the CRF stream.

n a positive integer chosen for the implementation

Copyright © 2016 IEEE. All rights reserved.

P_s the sample period of the CRF stream in nanoseconds

11. SDI Video Format

11.1 Overview

The SDI Video Format (SVF) is designated to carry direct mappings of the commonly used SDI (Serial Data Interface) data streams. It is a primary objective of the SVF to allow equipment originally designed to use one or more of these SMPTE SDI interface standards to migrate to, or additionally support, TSN-based transport of the SDI payload, with a minimum degree of added complexity or transcoding.

It is also a primary objective of the SVF to employ a packetization scheme that correlates directly with the horizontal scanning cadence of the SDI payload, such that a Listener of the SDI stream can easily decode the stream and utilize AVTP Presentation Time to reproduce the raster scanning as needed. This objective can also help an SVF Listener to identify a given SVF AVTPDU's position in the video raster, which enables synchronized switching between streams without inspecting the SVF AVTPDU's payload.

It is a secondary objective of the SVF to allow equipment intended to transport video streams across IP-based networks using the SMPTE ST 2022-6 standard to additionally support TSN-based transport of these signals over AVTP, where the synchronization and Quality-of-Service (QoS) mechanisms provided by gPTP and SRP are utilized.

The SMPTE ST 2022-6 standard defines a direct mapping of the SDI payload onto RTP-over-IP packets having a fixed RTP payload size of 1376 octets; however, it does not utilize gPTP timing information or correlate packet encoding to the horizontal scanning cadence of the SDI payload. Also, in SMPTE ST 2022-6 the usages of RTP timestamps and optional video timestamps are defined such that only the first media sample of an entire video frame has a known correlation to a specific location (timing point) in the video raster (see SMPTE ST 2022-6:2012 6.3 and 6.4).

11.2 SDI Video Stream data encapsulation

11.2.1 Overview

This encapsulation shall use the common stream header and shall have a **subtype** field of SVF (see 4.4.3.2).

This encapsulation uses the following common stream fields (see 4.4.3 and 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

— **stream_id**: 8 octets

Copyright © 2016 IEEE. All rights reserved.

Specific uses are given for the following common stream fields:

— **avtp_timestamp**: 4 octets

— **stream_data_length**: 2 octets

This encapsulation uses the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, **format_specific_data_3**, and **stream_data_payload** fields of the common stream header to define the following format-specific fields:

— **rsv** (reserved): 2 bits

— **reserved**: 7 bits

— **format**: 1 octet

— **i_seq_num**: 1 octet

— **line_number**: 2 octets

— **r** (reserved): 1 bit

— **gb** (guard band): 1 bit

— **sp** (RP168 switch): 1 bit

— **ef** (end frame): 1 bit

— **evt**: 4 bits

— **reserved**: 1 octet

The SDI Header is also added and contains the following fields:

— **map**: 4 bits

— **sample**: 4 bits

— **frame**: 1 octet

— **frate**: 1 octet

— **frcount**: 1 octet

— **video_data_payload**: 0 to *n* octets

These fields are shown in Figure 76.

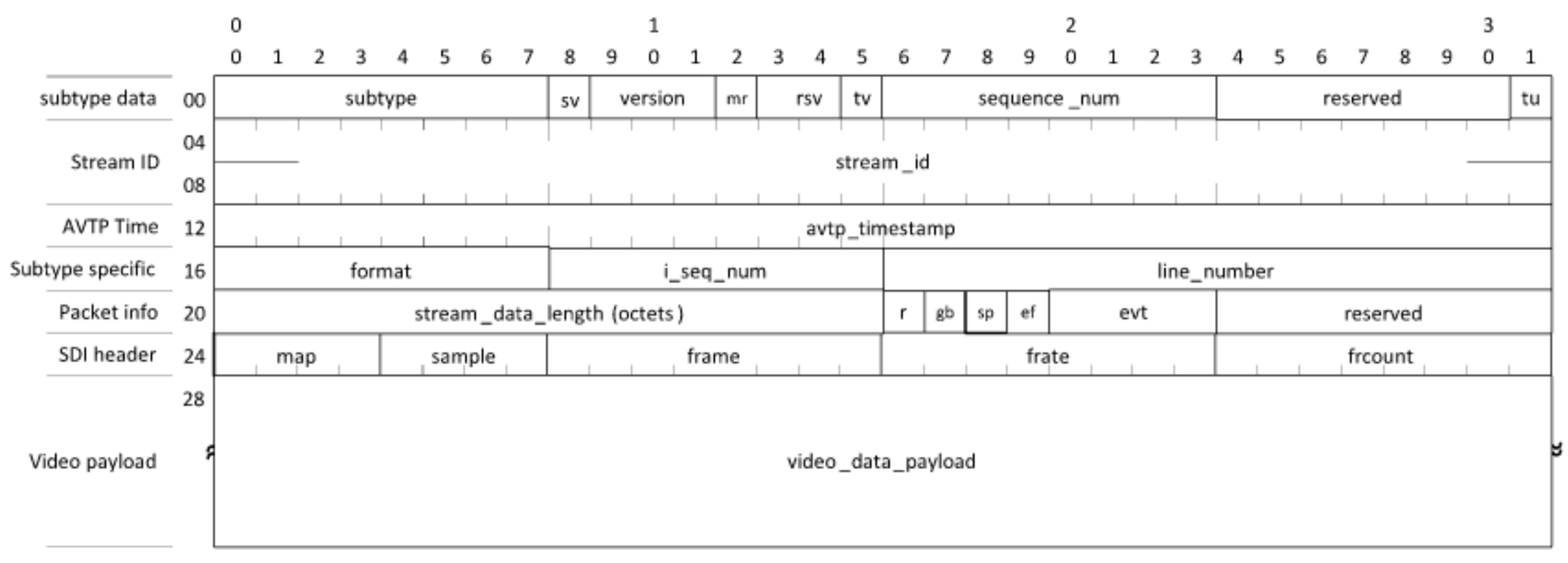


Figure 76—SDI Video Format PDU

Copyright © 2016 IEEE. All rights reserved.

11.2.2 avtp_timestamp field

A timestamp, as defined by 4.4.4.9, shall be included in the **avtp_timestamp** field on the first packet of each video line and the **tv** bit shall be set. On all other SVF AVTPDUs, **tv** shall be cleared.

This timestamp corresponds to the video line associated with the data in the SVF AVTPDU. If this is the first SVF AVTPDU for a new video line (when **tv** bit is set), the AVTP Presentation Time corresponds to the leading edge of horizontal blanking at the start of the video line. The SDI payload contains the code sequence for End of Active Video (EAV) in the first payload fields (four successive 10-bit data words) of each video line, and the first 10-bit word in this sequence represents the leading edge of horizontal blanking.

NOTE—As shown in Figure 77, an SDI-encoded video frame, as well as each individual line within the raster, contains both active video (image) data and non-image data (designated either as blanking or ancillary). Therefore, the **avtp_timestamp** does not always correspond to an active pixel of the video image; rather, the timestamp corresponds to the instant at which scanning of the current line begins. Each SDI-encoded video line also includes a code sequence for Start of Active Video (SAV) that separates the blanking and ancillary data from the image data. For the purposes of this standard (and to be consistent with legacy television scanning standards), only the EAV sequence (and not the SAV sequence) is used as a timing mechanism. The method used by the AVTP Listener to reproduce raster scanning based on the **avtp_timestamp** and the actual payload contents is outside the scope of this standard.

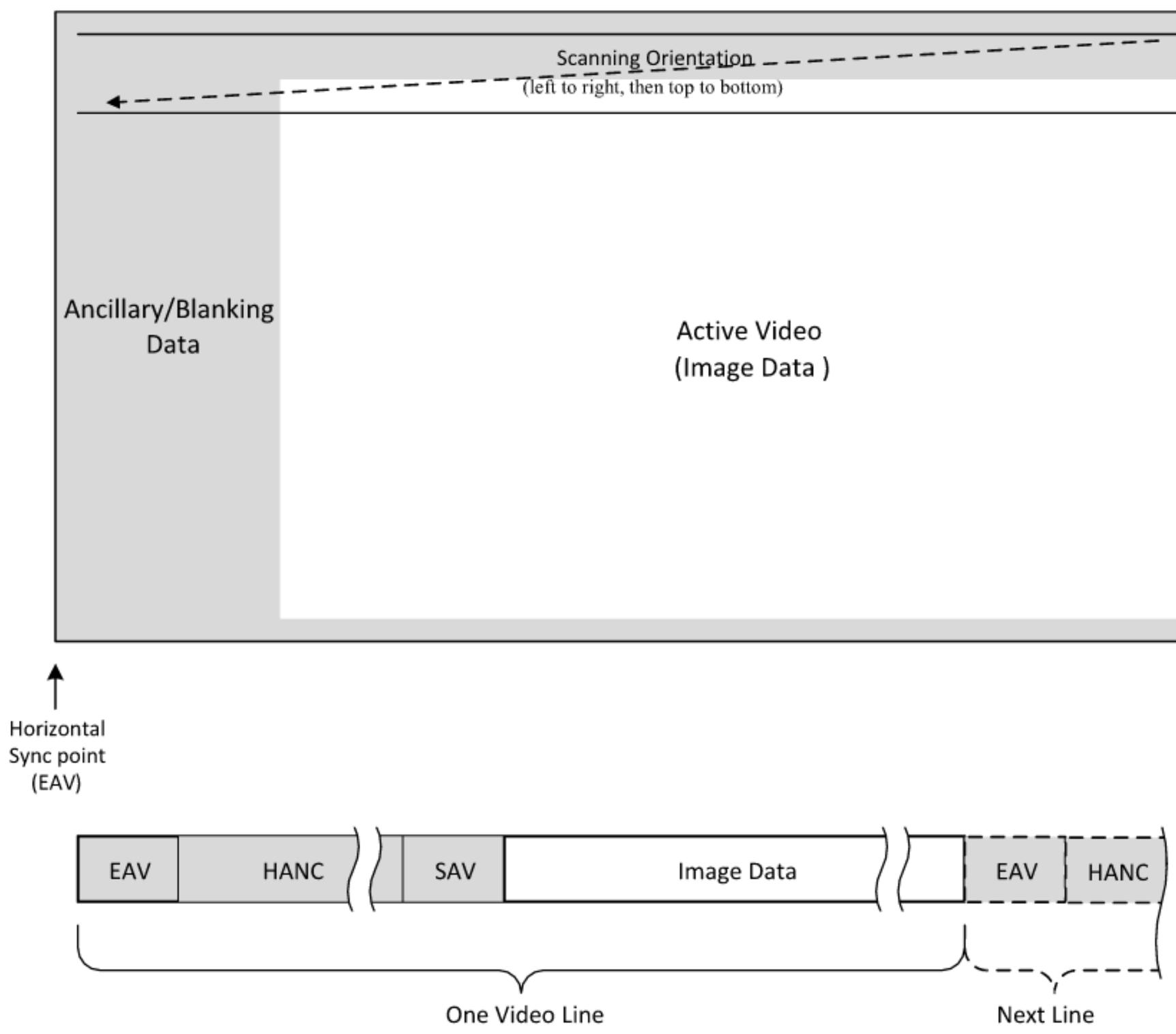


Figure 77 —Structure of SDI-encoded video frames and lines

Copyright © 2016 IEEE. All rights reserved.

11.2.3 format field

The **format** field is used to provide a simple method for determining the video source format contained in an AVTP stream, as specified in Table 32. SVF is primarily intended to support commonly used mainstream raster sizes and associated video frame rates. There is no expectation to define a unique **format** value for each possible permutation that might be deployed in uncommon applications or proprietary systems. In cases where a source format is used that is not given in Table 32, the **format** field is set to SVF_USER, indicating that the **map**, **sample**, **frame**, and **frate** fields define the desired source format.

Table 32—SVF format values for common raster structures

Value	Raster structure	Resolution type	SDI bitrate	Reference standard/system no.	Frame rate (Hz)	Scanning format
0_{16}	Reserved	—	—	—	—	—
1_{16}	525i/59.94	Standard definition	270 Mb/s	ST259, BT.656	$30 \times P_d$ [a]	Interlaced
2_{16}	625i/50	Standard definition	270 Mb/s	ST 259, BT.656	25	Interlaced
3_{16}	1080i/59.94	High definition	1.485 Gb/s	ST 274/5	$30 \times P_d$	Interlaced
4_{16}	1080i/50	High definition	1.485 Gb/s	ST 274/6	25	Interlaced
5_{16}	720p/59.94	High definition	1.485 Gb/s	ST 296/2	$60 \times P_d$	Progressive
6_{16}	720p/50	High definition	1.485 Gb/s	ST 296/3	50	Progressive
7_{16}	1080p/23.98	High definition	2.97 Gb/s	ST 274/11	$24 \times P_d$	Segmented frame
8_{16}	1080p/59.94	High definition	2.97 Gb/s	ST 274/2	$60 \times P_d$	Progressive
9_{16}	1080p/50	High definition	2.97 Gb/s	ST 274/3	50	Progressive
A to FE_{16}	Reserved	—	—	—	—	—
FF_{16}	SVF_USER	—	—	—	—	—

[a] Where P_d is a "pull-down" equal to 1000/1001. For example, $30 \times P_d = 29.97003$.

When the **format** value is set to SVF_USER, the SDI Header fields **map**, **sample**, **frame**, and **frate** shall be used to specify unique combinations of these video source format parameters.

When the **format** value is not set to SVF_USER, then the **map**, **sample**, **frame**, and **frate** fields shall be set to the values in Table 33 to avoid any possible mismatch between the values defined for the **format** and the values written explicitly in the SDI Header. When tshe values in Table 33 are used, an SVF Listener shall ignore the **map**, **sample**, **frame**, and **frate** fields on receive.

NOTE—The SDI Header fields **map**, **sample**, **frame**, and **frate** are used in this standard in the same way they are defined in SMPTE ST 2022-6. These fields facilitate support in this standard for various non-mainstream payload formats that may be used by an application supporting SMPTE ST 2022-6, i.e., those formats that correspond to a **format** of SVF_USER in this standard.

Copyright © 2016 IEEE. All rights reserved.

Table 33—SVF field values for defined format entries

Field	Value
map	F_{16}
sample	F_{16}
frame	FF_{16}
frate	FF_{16}

11.2.4 i_seq_num field

The **i_seq_num** field is an intra-line sequence number, and shall be set to the value N, where the current SVF AVTPDU is SVF AVTPDU number (N + 1) of the current video line. When **i_seq_num** is zero (0), the first data of the **video_payload_data** field is the first data of a video line.

11.2.5 line_number field

The **line_number** field indicates the line number of the data within the SVF AVTPDU in the context of the current video frame. All media samples in the SVF AVTPDU shall be from the same line. When the **line_number** field increments to a new value, the **i_seq_num** field shall be reset to zero (0). The **line_number** field shall remain set to the same value for all AVTPDUs in a given video line. The line numbering starts at one (1) for each new video frame and increments by one (1) each new video line. There is no difference in line-number counting for interlaced or progressive video formats when using SDI Video Format.

11.2.6 stream_data_length field

The **stream_data_length** contains the total number of octets in the SVF AVTPDU after the **format_specific_data_3** field, which includes the SDI Header and the **video_data_payload**. The **stream_data_length** for a given SDI stream shall maintain a constant size (in octets) over the duration of the stream's lifetime. The **stream_data_length** is fixed for each **format** and shall be set to the value corresponding to the **format**, as specified in Table 34.

Table 34—SDI stream_data_length values

format	stream_data_length
0_{16}	Reserved
1_{16}	1077
2_{16}	1084
3_{16}	1383
4_{16}	1324
5_{16}	1379
6_{16}	1242
7_{16}	1379
8_{16}	1379
9_{16}	1324
A to FF_{16}	Reserved

11.2.7 gb (guard band) field

Ancillary data is carried in the blanking interval portions of the video data payload. Some ancillary data packet types, such as Dolby E audio (as encoded by SMPTE ST 337M), may span multiple video frames and so it is not always safe to switch video sources on an arbitrary video frame boundary. For example, in

Copyright © 2016 IEEE. All rights reserved.

the case of Dolby E, this situation can occur if the video frame rate is 60 frames/second and the Dolby E frame rate is 30 frames/second.

The **gb** (guard band) bit is used as an identifier to mark an ancillary data sync point, referred to as a meta-frame. Setting the **sp** bit indicates to video-switching equipment that it is a proper point in time at which to execute a switchover (i.e., on a boundary between successive meta-frames).

The gb bit shall be set to one (1) for the last SVF AVTPDU of a meta-frame. Otherwise, **gb** bit shall be set to zero (0).

11.2.8 sp (RP168 switch) field

The **sp** (RP168 switch) bit is used as an identifier to indicate SMPTE RP 168 switch areas. If the **sp** bit is set, the first sample in the SVF AVTPDU is in the SMPTE RP 168 switch area. Otherwise, the **sp** bit is set to zero (0).

11.2.9 ef (end frame) field

The **ef** (end frame) shall be set to one (1) for the last SVF AVTPDU of a video frame. This allows a Listener to know in advance that the next SVF AVTPDU will be the start of a new video frame. For all other SVF AVTPDUs, **ef** shall be set to zero (0).

NOTE—This standard adopts the same convention used in SMPTE ST 2022-6 for marking the transition to a new video frame.

11.2.10 evt field

The **evt** field contains 4 bits that are reserved for upper-level protocols to indicate events in the stream. The specific use of these bits is outside the scope of this standard.

11.2.11 map field

The **map** field indicates the top-level structure of the data stream and shall be set to one of the values defined in Table 35.

Table 35—map field values

Value	Description
0_{16}	Direct sample structure as per SMPTE ST 292 or SMPTE ST 425 Level A.
1 to E_{16}	Reserved
F_{16}	Defined by **format**

NOTE—The “Level B” options given in SMPTE ST 2022-6 are not supported for Ethernet connections. Any Level B SDI streams will need to be converted to Level A streams before being transported over AVTP.

Copyright © 2016 IEEE. All rights reserved.

11.2.12 sample field

The **sample** field indicates the component pixel sampling structure and bit depth of the payload, and shall be set to one of the values given in Table 36.

Table 36—sample field values

Value	Sampling structure	Bit depth
0_{16}	Unknown/User specified	—
1_{16}	4:2:2	10 bits
2_{16}	4:4:4	10 bits
3_{16}	4:4:4:4	10 bits
4_{16}	Reserved	—
5_{16}	4:2:2	12 bits
6_{16}	4:4:4	12 bits
7_{16}	4:4:4:4	12 bits
8_{16}	4:2:2:4	12 bits
9 to E_{16}	Reserved	—
F_{16}	Defined by **format**	—

11.2.13 frame field

The **frame** field indicates the luminance active pixel structure of the payload, and shall be set to one of the values given in Table 37.

Table 37—frame field values

Value	Horizontal active	Vertical active	Vertical total	Sampling structure	Transport structure
0_{16}	Unknown/User specified frame structure	—	—	—	—
1 to F_{16}	Reserved	—	—	—	—
10_{16}	720	486	525	Interlace	Interlace
11_{16}	720	576	625	Interlace	Interlace
12_{16}	1280	720	750	Progressive	Progressive
13 to $1F_{16}$	Reserved	—	—	—	—
20_{16}	1920	1080	1125	Interlace	Interlace
21_{16}	1920	1080	1125	Progressive	Progressive
22_{16}	1920	1080	1125	Progressive	Interlace (Segmented)
23_{16}	2048	1080	1125	Progressive	Progressive
24_{16}	2048	1080	1125	Progressive	Interlace (Segmented)
25 to $2F_{16}$	Reserved	—	—	—	—
30_{16}	1280	720	750	Progressive	Progressive
31 to FE_{16}	Reserved	—	—	—	—
FF_{16}	Defined by **format**	—	—	—	—

11.2.14 frate field

The **frate** field indicates the video frame rate of the payload, and shall be set to one of the values given in Table 38.

Copyright © 2016 IEEE. All rights reserved.

Table 38 —frate field values

Value	Frame rate (Hz)
0 to 4_{16}	User specified (see SMPTE ST 2022-6)
5 to F_{16}	Reserved
10_{16}	60
11_{16}	60/1.001
12_{16}	50
13_{16}	Reserved
14_{16}	48
15_{16}	48/1.001
16_{16}	30
17_{16}	30/1.001
18_{16}	25
19_{16}	Reserved
$1A_{16}$	24
$1B_{16}$	24/1.001
1C to FE_{16}	Reserved
FF_{16}	Defined by **format**

11.2.15 frcount field

The **frcount** field identifies a video frame counter value. The counter shall increment by one (1) at the start of a video frame, which is the SVF AVTPDU following an SVF AVTPDU that has the **ef** field set.

A Listener may join a stream at any time after the stream has started, and therefore, a Listener shall be able to tolerate a starting **frcount** of any value.

11.2.16 video_data_payload field

The **video_data_payload** field contains the video sample data as defined by the **format** field. Specifically, for a given **format** the document listed in the Reference Standard column of Table 32 specifies the contents and formatting of the video sample data to be inserted in this field.

The **video_data_payload** field contains octets as defined by SMPTE ST 2022-6:2012 Section 6.5 "High Bit Rate Media Payload" with the exception that the payload includes samples of the luminance and color difference values as defined by the **format**, or alternatively by the **frame**, **frate**, and **sample** fields if the **format** is set to SVF_USER.

The number of octets in the **video_data_payload** field is static for all SVF AVTPDUs of a stream and is equal to (**stream_data_length** − 8).

When the SVF AVTPDU is the first SVF AVTPDU of a video line (**i_seq_num** is zero [0]), the first data in the **video_data_payload** field shall be the code for End of Active Video (EAV) as shown in Figure 77. For all other SVF AVTPDUs in the same stream, the first data in the **video_data_payload** is a continuation of the video line data from the previous SVF AVTPDU, such that no gaps in data exist between the last data of the **video_data_payload** field of the previous SVF AVTPDU and the first data of this field in the current SVF AVTPDU. Because the SDI-encoding standards use 10-bit or 12-bit sample words, it is possible that the continuation between successive SVF AVTPDUs may occur in the middle of a sample word in the payload's data.

Table 39 specifies video line information for each defined **format**. Video formats with **format** of SVF_USER may also be used, but the video line information is not given in this document.

Copyright © 2016 IEEE. All rights reserved.

— **SVF AVTPDUs per line**: the number of SVF AVTPDUs required to send a full video line of data

— **Video data size**: the number of octets containing video data for each SVF AVTPDU that is not the last SVF AVTPDU of a video line

— **Last SVF AVTPDU size**: the number of octets containing video data present in the last SVF AVTPDU of a video line (**ef** bit set)

— **Zero pad size**: the number of octets of zero padding that are added to the end of the **video_data_payload** only in the last SVF AVTPDU of a video line to meet the requirement that all SVF AVTPDUs for a given stream are the same size

Table 39—SVF payload specifics

Format	SVF AVTPDUs per line	Video data size	Last SVF AVTPDU size	Zero pad size
0_{16} (Reserved)	—	—	—	—
1_{16}	2	1073	1072	1
2_{16}	2	1080	1080	0
3_{16}	4	1375	1375	0
4_{16}	5	1320	1320	0
5_{16}	3	1375	1375	0
6_{16}	4	1238	1236	2
7_{16}	5	1375	1375	0
8_{16}	4	1375	1375	0
9_{16}	5	1320	1320	0
A to FE_{16} (Reserved)	—	—	—	—
FF_{16}	Not defined	Not defined	Not defined	Not defined

12. Raw Video Format

12.1 Overview

The Raw Video Format (RVF) is designed to carry direct mappings of raw video data streams. It is a primary objective of RVF to allow all kinds of video rasters to support TSN-based transport of raw video payloads, with a minimum degree of added complexity or transcoding. It is also a primary objective of RVF to employ a packetization scheme that correlates directly with the horizontal scanning cadence of the raw video payload, such that a Listener of the RVF stream can easily decode the stream and utilize AVTP Presentation Time to reproduce the raster scanning as needed. This objective can also help an RVF Listener identify a given RVF AVTPDU's position in the video raster, which may enable synchronized switching between streams without inspecting the RVF AVTPDU's payload.

It is a secondary objective of RVF to allow the addition of blanking lines for carrying non-video data in an RVF AVTPDU. In case of adding blanking lines, only vertical blanking is allowed. No horizontal blanking can be transmitted using RVF.

It is a tertiary objective of RVF to support the packetization of computer graphics.

Copyright © 2016 IEEE. All rights reserved.

12.2 Raw Video Stream data encapsulation

12.2.1 Overview

The Raw Video Stream data encapsulation is used for carrying video streams defined by this clause.

This encapsulation shall use the common stream header and shall have a **subtype** field of RVF (see 4.4.3.2).

This encapsulation uses the following common stream fields (see 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

— **stream_id**: 8 octets

Specific uses are given for the following common stream fields:

— **avtp_timestamp**: 4 octets

This encapsulation uses the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, **format_specific_data_3**, and **stream_data_payload** fields of the common stream header to define the following format-specific fields:

— **rsv**: 2 bits

— **reserved**: 7 bits

— **active_pixels**: 2 octets

— **total_lines**: 2 octets

— **stream_data_length**: 2 octets

— **ap** (active pixels): 1 bit

— **r** (reserved): 1 bit

— **f** (field): 1 bit

— **ef** (end frame): 1 bit

— **evt**: 4 bits

— **pd** (pull-down): 1 bit

— **i** (interlaced): 1 bit

— **reserved**: 6 bits

The **stream_data_payload** field of the Common Stream Header is defined to contain the following fields:

— **reserved**: 1 octet

Copyright © 2016 IEEE. All rights reserved.

— **pixel_depth**: 4 bits
— **pixel_format**: 4 bits
— **frame_rate**: 1 octet
— **colorspace**: 4 bits
— **num_lines**: 4 bits
— **reserved**: 1 octet
— **i_seq_num**: 1 octet
— **line_number**: 2 octets
— **video_data_payload**: 0 to *n* octets

These fields are shown in Figure 78.

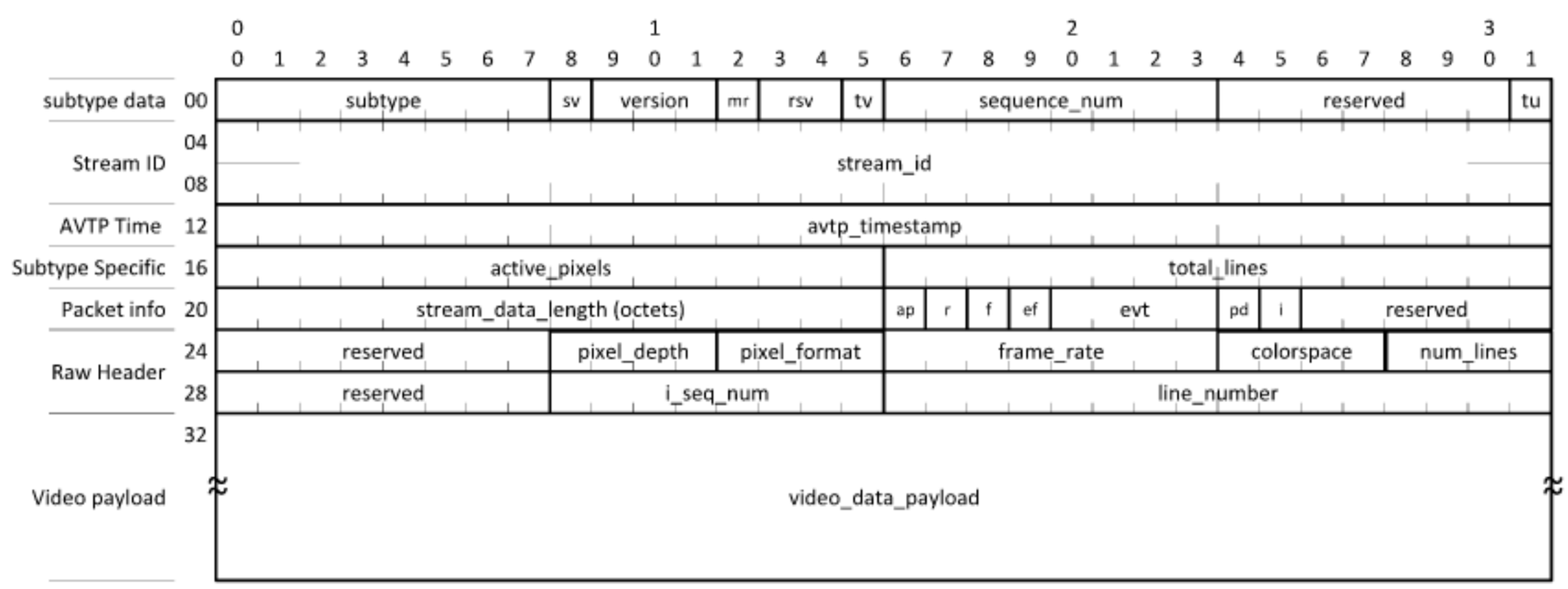


Figure 78—Raw Video PDU Format

12.2.2 avtp_timestamp field

If **num_lines** (12.2.15) is zero (0) then a timestamp, as defined by 4.4.4.9, shall be included in the **avtp_timestamp** field on every RVF AVTPDU for which **i_seq_num** (12.2.16) set to zero (0). When this condition is met, the **tv** bit shall also be set. On all other RVF AVTPDUs, **tv** shall be cleared.

When **num_lines lines** (12.2.15) is greater than zero (0) a timestamp, as defined by 4.4.4.9, shall be included in the **avtp_timestamp** field on every RVF AVTPDU for which **line_number** (12.2.17) is set to one (1). When this condition is met, the **tv** bit shall also be set. On all other RVF AVTPDUs, **tv** shall be cleared.

This timestamp corresponds to the AVTP Presentation Time of the first sample of data in the RVF AVTPDU. If this is the first RVF AVTPDU for a new video line (when **tv** bit is set), the AVTP Presentation Time corresponds to the first sample of transmitted data of that video line.

12.2.3 active_pixels field

The **active_pixels** field indicates the number of active pixels in a video line. The number of active pixels shall be the same for every line and shall not change through the life of the stream.

Copyright © 2016 IEEE. All rights reserved.

12.2.4 total_lines field

The **total_lines** field indicates the total number of lines of the video frame. For interlaced formats, the total number of lines should be the total of both video fields. The total number of lines can be higher than the active video lines to include vertical blanking lines.

12.2.5 ap (active pixels) field

The **ap** (active pixels) bit is used as an identifier to mark the video lines that contain active pixels. The **ap** bit is set to one (1) for all RVF AVTPDUs that contain active area pixels. The **ap** bit is set to zero (0) for all RVF AVTPDUs that contain pixels outside the active area (see Figure 79).

Each RVF AVTPDU shall have either all active pixels (**ap** is set) or all vertical blanking pixels (**ap** is not set). A mixture of active and blanking lines within the same RVF AVTPDU is not permitted.

12.2.6 f (field) field

The **f** (field) field is used with interlaced video payload formats to distinguish between the first and second fields in an interlaced pair. The **f** field shall be set to zero (0) for all RVF AVTPDUs of Field 1, and **f** shall be set to one (1) for all RVF AVTPDUs of Field 2. If the video format is non-interlaced, **f** shall always be zero (0).

12.2.7 ef (end frame) field

The **ef** (end frame) field shall be set to one (1) for the last RVF AVTPDU of a video frame. This allows a Listener to know in advance that the next RVF AVTPDU will be the start of a new video frame. For all other RVF AVTPDUs, **ef** shall be set to zero (0).

12.2.8 evt field

The **evt** field contains 4 bits that are reserved for upper-level protocols to indicate events in the stream. The specific use of these bits is outside the scope of this standard.

12.2.9 pd (pull-down) field

The **pd** (pull-down) field is used to indicate a pull-down of the video frame rate. When **pd** is set to one (1), a multiplier ratio of 1000/1001 is applied to the frame rate specified by the **frame_rate** (12.2.13) field. When **pd** is zero (0) there is no pull-down applied to the frame rate (i.e., a 1/1 ratio). For example when **pd** is set to one (1) and the video frame rate is 30 Hz, the frame rate becomes 30 Hz × 1000/1001 = 29.97003 Hz.

12.2.10 i (interlaced) field

The **i** (interlaced) field is used to indicate that the video source is an interlaced video format. When **i** is set to one (1) the payload data is interlaced. When **i** is set to zero (0), the payload data is non-interlaced (progressive).

Copyright © 2016 IEEE. All rights reserved.

12.2.11 pixel_depth field

The **pixel_depth** field indicates the number of bits allocated for each pixel and shall be set to one of the values given in Table 40.

Table 40 —pixel_depth values

Value	Pixel depth (bits)
0_{16}	Reserved
1_{16}	8
2_{16}	10
3_{16}	12
4_{16}	16
5 to E_{16}	Reserved
F_{16}	User defined

12.2.12 pixel_format field

The **pixel_format** field indicates the component pixel sampling structure of the payload, and shall be set to one of the values given in Table 41.

Table 41 —pixel_format values

Value	Pixel format
0_{16}	Monochrome
1_{16}	4:1:1
2_{16}	4:2:0
3_{16}	4:2:2
4_{16}	4:4:4
5_{16}	Reserved
6_{16}	4:2:2:4
7_{16}	4:4:4:4
8_{16}	Bayer grbg
9_{16}	Bayer rggb
A_{16}	Bayer bggr
B_{16}	Bayer gbrg
C to E_{16}	Reserved
F_{16}	User defined

12.2.13 frame_rate field

The **frame_rate** field indicates the video frame rate of the payload, and shall be set to one of the values given in Table 42.

Copyright © 2016 IEEE. All rights reserved.

Table 42—frame_rate field values

Value	Video frame rate (Hz)
0_{16}	Reserved
1_{16}	1
2_{16}	2
3_{16}	5
4 to F_{16}	Reserved
10_{16}	10
11_{16}	15
12_{16}	20
13_{16}	24
14_{16}	25
15_{16}	30
16_{16}	48
17_{16}	50
18_{16}	60
19_{16}	72
$1A_{16}$	85
1B to $2F_{16}$	Reserved
30_{16}	100
31_{16}	120
32_{16}	150
33_{16}	200
34_{16}	240
35_{16}	300
36 to FE_{16}	Reserved
F_{16}	User defined

12.2.14 colorspace field

The **colorspace** field indicates the top-level structure of the data stream and shall be set to one of the values defined in Table 43.

Table 43—colorspace field values

Value	Description
0_{16}	Reserved
1_{16}	YCbCr [B19], [B21], [B22]
2_{16}	sRGB [B10]
3_{16}	YCgCo
4_{16}	Grayscale
5_{16}	XYZ [B14]
6_{16}	YCM
7_{16}	BT Rec.601 [B19]
8_{16}	BT Rec.709 [B21]
9_{16}	ITU BT 2020 [B22]
A to E_{16}	Reserved
F_{16}	User defined

12.2.15 num_lines field

The **num_lines** field shall be set to the number of full lines contained in the RVF AVTPDU. If the RVF AVTPDU contains only a portion of a line, then the **num_lines** field shall be set to zero (0).

Copyright © 2016 IEEE. All rights reserved.

12.2.16 i_seq_num field

The **i_seq_num** field is an intra-line sequence number, and shall be set to the value N, where the current RVF AVTPDU is RVF AVTPDU number (N + 1) of the current video line. When **i_seq_num** is zero (0), the first data of the **video_payload_data** field is the first data of a video line.

Since an RVF AVTPDU containing multiple lines carries only full lines, if **num_lines** is not set to zero (0), then the **i_seq_num** field shall be set to zero (0).

12.2.17 line_number field

The **line_number** field contains the line number of the first line, or portion of a line, contained in the RVF AVTPDU.

If the RVF AVTPDU contains multiple full lines (the **num_lines** field is nonzero), then the **line_number** field shall be set to the line number of the first line in the RVF AVTPDU. The **line_number** starts at one (1) for each new video frame and increments by **num_lines** in each successive RVF AVTPDU.

If the RVF AVTPDU contains a partial line (the **num_lines** field is zero [0]), then the **line_number** field is set to the line number of the current partial line. When the **line_number** field increments to a new value, the **i_seq_num** field is reset to zero (0). The **line_number** field is set to the same value for all RVF AVTPDUs in a given video line. The line numbering starts at one (1) for each new video frame and increments by one (1) on each new video line.

There is no difference in line-number counting for interlaced or progressive video formats when using RVF.

12.2.18 RVF payload encapsulation

12.2.18.1 Encapsulation format

RVF is primarily intended for transmitting only the active part of a video frame. It also provides an option to extend the active area with additional vertical blanking lines. These additional lines can be used to preserve ancillary data in the vertical blanking. It can be more flexible to transmit the ancillary data separate from the active video, but for ease of transition from full SDI video to raw video, embedding ancillary data within the vertical blanking of an RVF stream is possible.

Transmission of horizontal blanking is not permitted in RVF. In case of interlaced video, this is Field 1 and Field 2 without the blanking in between or the horizontal blanking in front of an active line. In case of progressive video, the active area is one (1) complete video frame, no fields. In Figure 79 a representation of an interlaced video frame is shown.

Copyright © 2016 IEEE. All rights reserved.

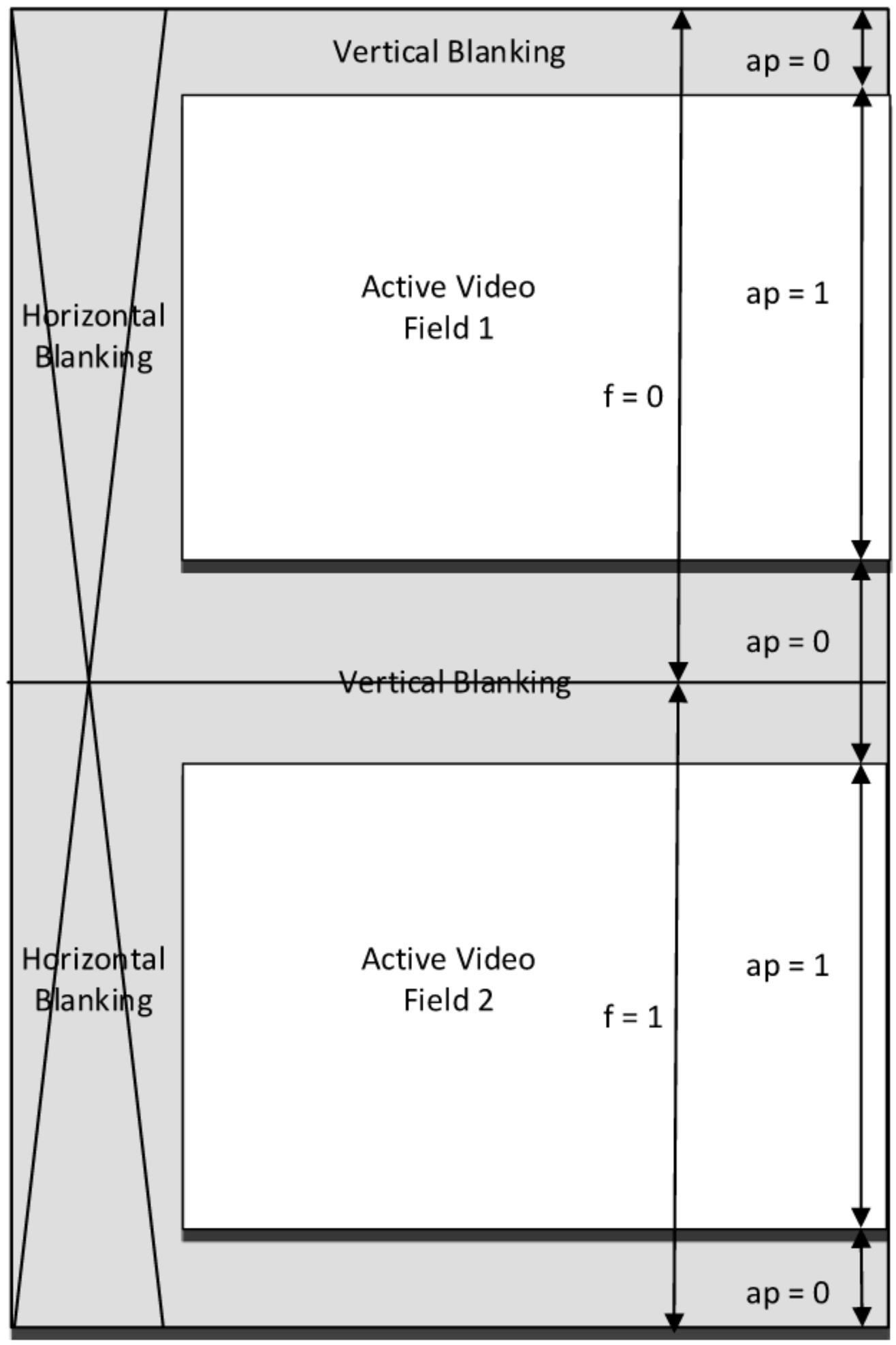


Figure 79—Structure of raw-encoded interlaced video raster

In Figure 80 a representation of a progressive video frame is shown.

Copyright © 2016 IEEE. All rights reserved.

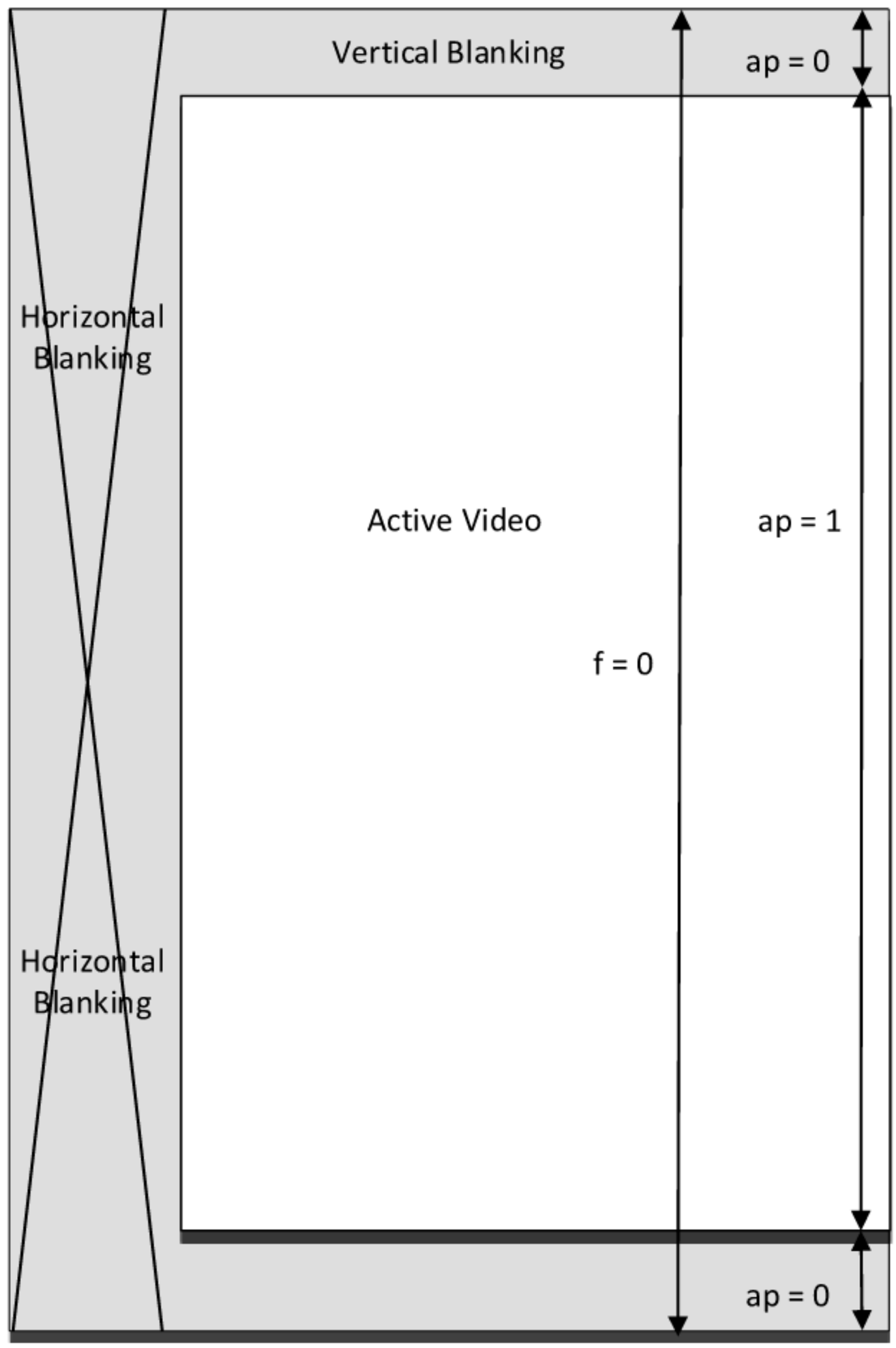


Figure 80—Structure of raw-encoded progressive video raster

Refer to 12.2.5 and 12.2.6 for the definitions of **ap** and **f**, respectively.

12.2.19 video_data_payload field for large rasters

For video rasters where the line size is long (**num_lines** is zero [0]), the first RVF AVTPDU of a video line (**i_seq_num** is zero [0]) starts with the first active pixel of that line. For all other RVF AVTPDUs in the same stream, the first data in the **video_data_payload** is a continuation of the video line data from the previous RVF AVTPDU, such that no gaps in data exist between the last data of the **video_data_payload** field of the previous RVF AVTPDU and the first data of this field in the current RVF AVTPDU. Note that because the raw-encoding standards may use 10-bit or 12-bit sample words, it is possible that the

Copyright © 2016 IEEE. All rights reserved.

continuation between successive RVF AVTPDUs may occur in the middle of a sample word in the payload's data.

All RVF AVTPDUs for a given stream are the same size. The **video_data_payload** size is calculated using the formula in Equation (17). The last RVF AVTPDU of a frame is padded if needed after the last pixel of that line to so that the **video_data_payload** is always the same size.

$$\left\lceil \frac{S \times B \times A}{8 \times P} \right\rceil \leq M \tag{17}$$

where

S sample words per pixel

B bits per sample word

A active pixels per line

P packets per line; lowest integer value to meet criteria

M max payload size, integer number of octets

$\lceil x \rceil$ the brackets represent the ceiling function

Table 44 gives examples of assembling the payload data for a few specific raster formats, with column definitions given here:

— **active pixels**: the number of pixels in a video line of data

— **samples per words**: number of samples for a complete word

— **pixel depth**: number of bits per pixel

— **RVF AVTPDUs per Line**: the minimum number of RVF AVTPDUs required to send a full video line of data

— **payload size**: the number of octets for each RVF AVTPDU.

Table 44—RVF Payload Examples (large raster size)

Active pixels	Sample words per pixel	Pixel depth	RVF AVTPDUs per line	Payload size
720	2	10	2	900
1280	2	10	3	1067
1920	2	10	4	1200
3840	2	10	7	1372

12.2.20 video_data_payload field for small rasters

For video rasters where the line size is short (**num_lines** is nonzero), each RVF AVTPDU contains one or more full video lines, where the number of lines is indicated by the **num_lines** field. The **i_seq_num** field is always zero (0).

Copyright © 2016 IEEE. All rights reserved.

The **video_data_payload** size is calculated using the formula in Equation (18).

$$\left\lceil \frac{S \times B \times A \times L}{8} \right\rceil \leq M \tag{18}$$

where

S sample words per pixel

B bits per sample word

A active pixels per line

L lines per packet, integer number of lines

M max payload size, integer number of octets

$\lceil x \rceil$ the brackets represent the ceiling function

Table 45 gives examples of assembling the payload data for a few specific raster formats, with column definitions given here:

— **active pixels**: the number of pixels in a video line of data

— **samples per words**: number of samples for a complete word

— **pixel depth**: number of bits per pixel

— **lines per RVF AVTPDU**: the number of lines that are transmitted in each RVF AVTPDU

— **payload size**: the number of octets for each AVTPDU.

Table 45—RVF payload examples (small raster size)

Active pixels	Sample words per pixel	Pixel depth	Lines per RVF AVTPDU	Payload size
400	1	10	2	1000
360	1	10	2	900
320	1	12	2	960
240	2	8	2	960
160	2	10	2	800
64	2	10	6	960

13. AES Encrypted Format

13.1 Overview

The AES Encrypted Format (AEF) is an encapsulated AVTPDU that provides a mechanism for carrying a payload encrypted with AES-SIV cryptography as defined in RFC 5297.

Copyright © 2016 IEEE. All rights reserved.

13.2 AEF data encapsulation

13.2.1 Overview

This subclause documents the fields contained in an AVTPDU when the **subtype** field is AEF_CONTINUOUS or AEF_DISCRETE (See 4.4.3.2):

— The **subtype** field shall be AEF_CONTINUOUS when the encapsulated AVTPDU has an encapsulation style of Continuous (Table 6).

— The **subtype** field shall be AEF_DISCRETE when the encapsulated AVTPDU has an encapsulation style of Discrete (Table 6).

This encapsulation uses the alternative header.

The **h** (header_specific) bit (4.4.3.3) of the common header is used to define the following:

— **r** (reserved) field: 1 bit

This encapsulation uses the following alternative header field directly (see 4.4.6 for definitions):

— **version** field: 3 bits

This encapsulation uses the **alternative_data_payload** field to define the following fields:

— **enc** field: 4 bits

— **reserved** field: 1 octet

— **blocks** field: 1 octet

— **key_id** field: 8 octets

— **encrypted_payload** field: 0 to *n* octets

Figure 81 shows the AEF AVTPDU for AEF_CONTINUOUS and AEF_DISCRETE subtypes.

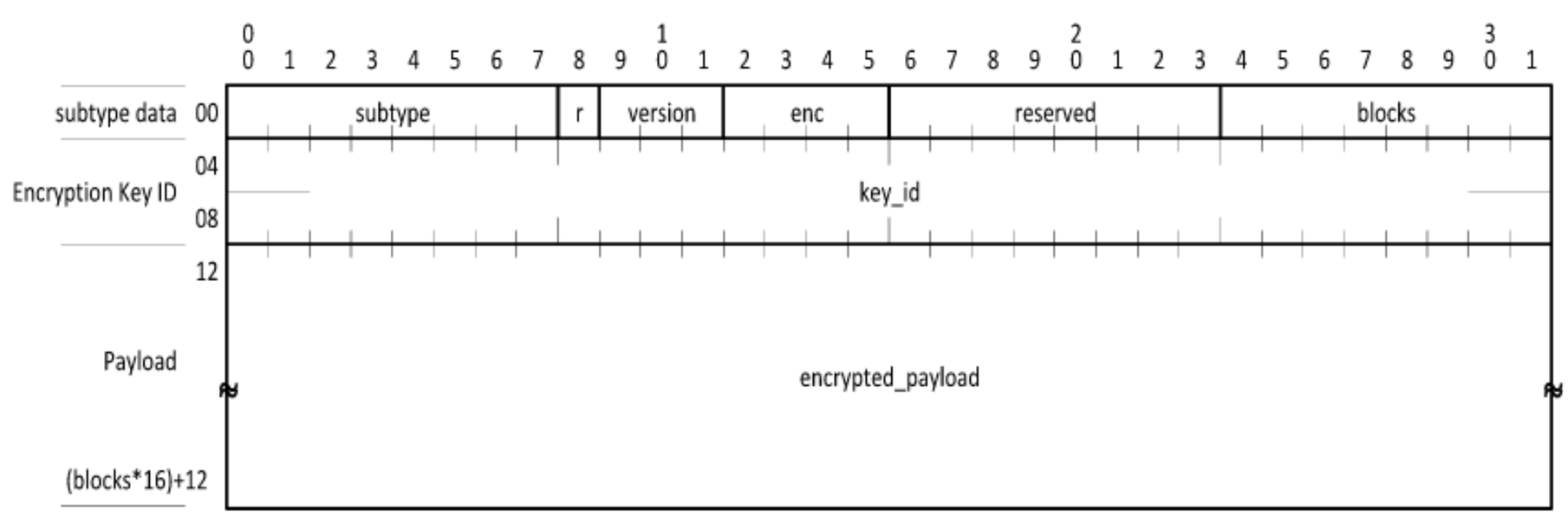


Figure 81 —Header for AES encrypted AVTPDU payload (AEF_CONTINUOUS and AEF_DISCRETE subtypes)

Copyright © 2016 IEEE. All rights reserved.

13.2.2 enc field

The **enc** field specifies the encryption format. The value of the **enc** field is defined in Table 46.

Table 46—enc field values

Value	Name	Description	Reference
0_{16}	AES_SIV	AES-SIV encryption	Subclause 13.3
1 to F_{16}	—	Reserved	—

13.2.3 blocks field

The **blocks** field specifies the payload length in AES blocks. One AES block is 128 bits, or 16 octets. The **blocks** field shall have a minimum value of one (1) and a maximum value of $5D_{16}$.

13.2.4 key_id field

The **key_id** field specifies an identifier that identifies a specific AES-128 or AES-256 key that is shared between the Talker and the intended Listeners of the AEF AVTPDU.

13.2.5 encrypted_payload field

The **encrypted_payload** field contains the output of the AES encryption algorithm. The **encrypted_payload** field shall have a length that is an integer multiple of 16 octets.

13.3 AES-SIV

13.3.1 Overview

The payload encapsulated in an AEF AVTPDU is a complete AVTPDU encrypted with the AES-SIV cryptographic algorithm as defined in RFC 5297.

As the input to the AES-SIV algorithm is 128-bit (16-octet) blocks of data, the encapsulated AVTPDU is padded at the end with additional octets with a value of zero (0) in order to make the length an integer multiple of 16 octets.

Encapsulated AVTPDUs are required to fit within the **encrypted_payload** field and be a multiple of 16 octets in length, and as such have a reduction in their maximum length. Additionally, the encryption process adds a further 16 octets of data. Therefore, the encapsulated AVTPDU's maximum length (EncapMTU) is 1472 octets as calculated by the following equation:

$$\text{EncapMTU} = \text{MTU} - \text{subtype_data}_{\text{length}} - \text{EncryptionKeyId}_{\text{length}} - 16 \tag{19}$$

When the encapsulated AVTPDU is decrypted, the padding will be present in the decrypted AVTPDU; however, the true data length of the encapsulated AVTPDU will be available within the encapsulated AVTPDU.

Copyright © 2016 IEEE. All rights reserved.

13.3.2 AES-SIV encryption

For AES-SIV encryption, the Talker uses the algorithm described in RFC 5297 Section 2.6.

The inputs to the algorithm are:

— K, the key value, which is the AES-256 private key that the **key_id** refers to

— P, the plaintext value, which is the encapsulated AVTPDU padded with zero (0) so that the length of P is a multiple of 16 octets

The output of the algorithm is:

— C, the ciphertext value that is used as the **encrypted_payload** field

The Talker sets the **blocks** field to the length of C in blocks of 16 octets.

13.3.3 AES-SIV decryption

For AES-SIV decryption, the Listener uses the algorithm described in RFC 5297 Section 2.7.

The inputs to the algorithm are:

— K, the key value, which is the AES-256 private key that the **key_id** refers to

— C, the cipher text, which is the contents of the **encrypted_payload** field in the AVTPDU

If the AES-SIV algorithm indicates authenticity, then the output of the algorithm is:

— P, the plaintext value, which is the encapsulated AVTPDU that will be padded up to be a multiple of 16 octets in length

— The authenticity of the AVTP frame

If the AES-SIV algorithm indicates the AVTP frame is not authentic, then the Listener shall discard this AEF AVTPDU.

14. Vendor Specific Format

14.1 Overview

The Vendor Specific Format (VSF) Stream Format, VSF_STREAM, is provided to allow vendors to define and use proprietary stream formats without causing interoperability issues with AVTP streams using formats defined elsewhere in this standard. VSF streams contain a unique identifier that is assigned by the organization that developed the format.

NOTE—There is no Vendor Specific Format defined for control formats. It is recommended that vendors use the General Purpose Control Message for this purpose, see 9.4.7.

Copyright © 2016 IEEE. All rights reserved.

14.1.1 Vendor specific stream data encapsulation

14.1.1.1 Overview

This encapsulation shall use the common stream header and shall have a **subtype** of VSF_STREAM (4.4.3.2).

This encapsulation uses the following common stream fields (see 4.4.3 and 4.4.4 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **mr**: 1 bit

— **tv**: 1 bit

— **sequence_num**: 1 octet

— **tu**: 1 bit

— **stream_id**: 8 octets

— **avtp_timestamp**: 4 octets

— **stream_data_length**: 2 octets

— **stream_data_payload**: 0 to *n* octets

This encapsulation uses the **f_s_d**, **format_specific_data_1**, **format_specific_data_2**, and **format_specific_data_3** fields of the common stream header to define the following format-specific fields:

— **rsv** (reserved): 2 bits

— **reserved**: 7 bits

— **vendor_id_1**: 4 octets

— **vendor_id_2**: 2 octets

These fields are shown in Figure 82.

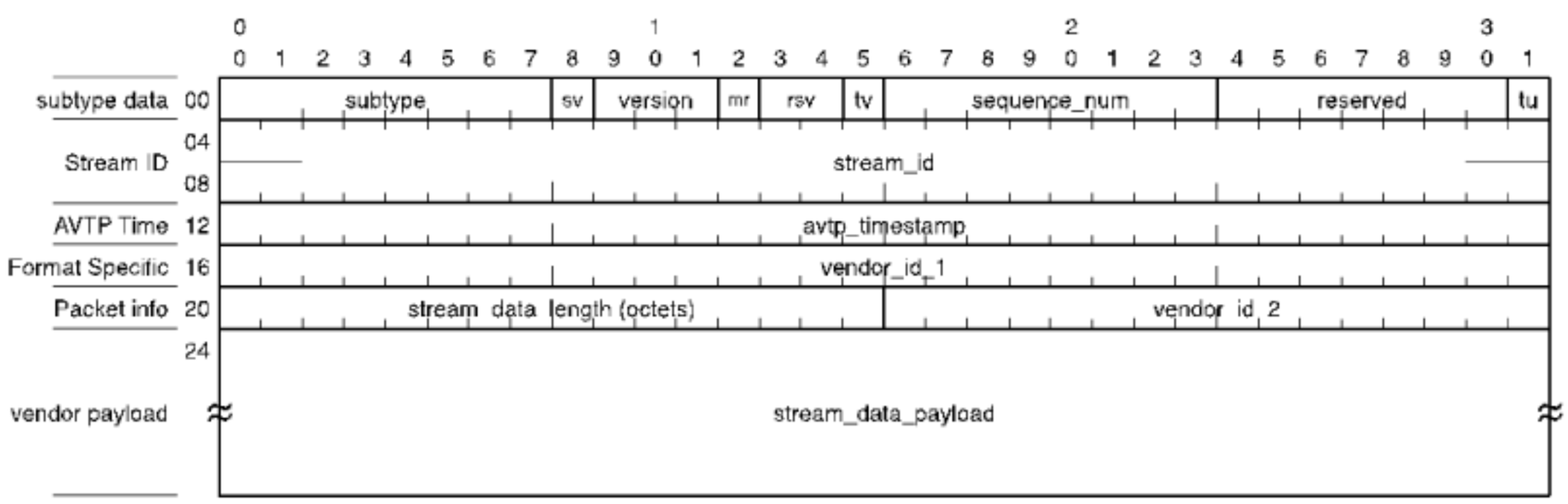


Figure 82—Vendor Specific PDU Format

Copyright © 2016 IEEE. All rights reserved.

14.1.1.2 vendor_id_1 and vendor_id_2 fields

The **vendor_id_1** and **vendor_id_2** fields shall contain a unique value assigned by the vendor based on the organization's MAC Address Block (MA-L [OUI], MA-M, or MA-S) or CID. There are two options for constructing this value. First, the fields could contain an EUI-48 in the AVTP Vendor Specific Format space[5] assigned by the vendor that developed the stream format using their own MA-L (OUI), MA-M, or MA-S, followed by a unique value that fills out the 48-bit field. Alternatively, the first octets contain the organization's CID, followed by a unique value that fills out the 48-bit field assigned by the organization holding the CID.[6]

The unique value of the vendor ID is split across the fields such that the most-significant 4 octets (bits 0 to 31) are contained in the **vendor_id_1** field and the least-significant 2 octets (bits 32 to 47) are in the **vendor_id_2** field.

14.1.1.3 stream_data_payload field

The **stream_data_payload** field contains the vendor-specific stream data.

15. Experimental Format

15.1 Overview

The experimental stream (EF_STREAM) and control (EF_CONTROL) subtypes are provided as a mechanism for testing and developing future enhancements to the IEEE 1722 family of standards.

The **subtype** values of EF_STREAM and EF_CONTROL shall not be used in a conforming product.

Experimental stream and control formats may define the layout of all fields following the AVTP common header (4.4.3). When developing a format using one of the experimental **subtype** values, it is recommended that the layout and contents of the header follow those of the AVTP common control header (4.4.5) or AVTP common stream header (4.4.4).

16. ECC Signed Control Format

16.1 Overview

The ECC Signed Control Format (ESCF) is used to transport an encapsulated AVTPDU that is cryptographically signed with an Elliptic Curve Cryptography (ECC) algorithm as defined in IEEE Std 1363a™. The signing of an AVTPDU allows a Listener to verify the sender of the AVTPDU and also allows the Listener to verify the integrity of the encapsulated AVTPDU.

[5]The AVTP Vendor Specific Format space is the EUI-48 set that is used for identifying AVTP Vendor Specific Formats. This consists of all valid EUI-48s. An EUI-48 that is used to identify a Vendor Specific Format can be re-used with a different meaning in another EUI-48 space.
[6]https://standards.ieee.org/develop/regauth/tut/eui.pdf provides information on OUIs and CIDs.

Copyright © 2016 IEEE. All rights reserved.

16.2 ECC Signed Control Data encapsulation

16.2.1 Overview

This encapsulation shall use the alternative header and shall have a **subtype** field of ESCF (see 4.4.3.2).

The **h** (header_specific) bit (4.4.3.3) of the common header is used to define the following:

— **r** field: 1 bits

This encapsulation uses the following alternative header field directly (see 4.4.6 for definitions):

— **version** field: 3 bits

This encapsulation uses the **alternative_data_payload** field to define the following fields:

— **sig**: 4 bits
— **reserved** (reserved): 5 bits
— **encapsulated_avtpdu_length**: 11 bits
— **key_id**: 8 octets
— **signing_timestamp**: 8 octets
— **ecdsa_signature_c**: 32 octets
— **ecdsa_signature_d**: 32 octets
— **encapsulated_avtpdu**: variable length, from 12 octets to 1416 octets inclusive

These fields are shown in Figure 83.

Copyright © 2016 IEEE. All rights reserved.

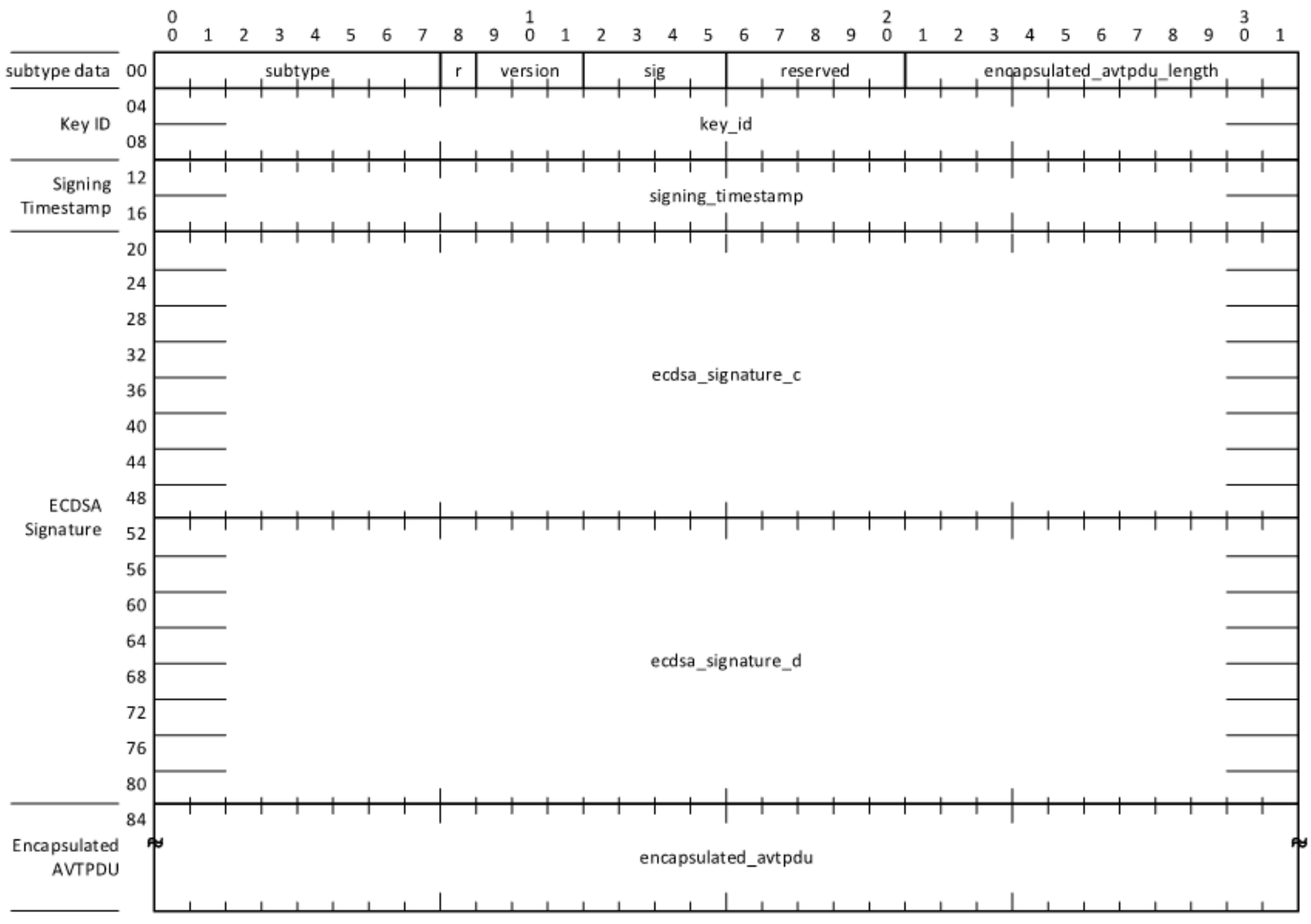


Figure 83—ECC_SIGNED_CONTROL AVTPDU

16.2.2 sig field

The **sig** field defines the ECC signature algorithm variant used by the ESCF AVTPDU.

The value of the **sig** field is defined in Table 47.

Table 47—sig field values

Value	Name	Description	Reference
0_{16}	ECSSA_DSA_EMSA1_SHA256	Elliptic curve signature scheme with appendix using ECSP-DSA, ECVP-DSA, EMSA1, and SHA-256	Subclause 16.3
1 to F_{16}	—	Reserved	—

16.2.3 encapsulated_avtpdu_length field

The **encapsulated_avtpdu_length** field specifies the length of the **encapsulated_avtpdu** field in octets.

16.2.4 key_id field

The 8-octet (64-bit) **key_id** field is an EUI-64 used to identify the elliptic curve private key used to sign the **encapsulated_avtpdu** field.

Copyright © 2016 IEEE. All rights reserved.

NOTE—The specific mechanism of how EUI-64 values are defined and assigned is outside the scope of this standard. For more information see subclause 7.6.1.1 in IEEE Std 1722.1-2013.

16.2.5 signing_timestamp field

The **signing_timestamp** field is an 8-octet field representing gPTP time in nanoseconds at the time that the signing algorithm was applied.

The **signing_timestamp** is constructed identically to the **CRF_timestamp** as defined in 10.4.13 and calculated in Equation (13).

16.2.6 encapsulated_avtpdu

The **encapsulated_avtpdu** field contains a complete AVTPDU starting at the **subtype** field. The **encapsulated_avtpdu** is limited in size to 1416 octets.

16.2.7 ecdsa_signature_c and ecdsa_signature_d fields

Both the **ecdsa_signature_c** and **ecdsa_signature_d** fields are 256-bit fields. The values in these fields are calculated by the signature variant algorithm that is selected via the value of the **sig** field.

16.3 ECSSA DSA EMSA1 SHA256 signature variant

16.3.1 Overview

When the **sig** field is set to ECSSA_DSA_EMSA1_SHA256, the ESCF AVTPDUs **ecdsa_signature_c** and **ecdsa_signature_d** fields correspond to the c and d values as calculated by using the ECSSA (subclause 10.2 of IEEE Std 1363a-2004) utilizing ECSP-DSA (subclause 7.2.7 of IEEE Std 1363a-2004) and ECVP-DSA (subclause 7.2.8 of IEEE Std 1363a-2004) with the EMSA1 (subclause 12.1.1 of IEEE Std 1363a-2004) message encoding method using a SHA-256 hash (subclause 14.1.3 of IEEE Std 1363a-2004).

The input to the SHA-256 hash is the concatenation of the following ESCF AVTPDU's fields:

— **key_id**

— **signing_timestamp**

— **encapsulated_avtpdu**

16.3.2 Signature generation

Generation of the signature of the AVTP frame is achieved by using the private key that is identified by the **key_id** with the signature generation operation defined in subclause 10.2.2 of IEEE Std 1363a-2004.

16.3.3 Signature validation

Validation of the signature of the AVTP frame is achieved by using the public key that is identified by the **key_id** with the signature verification operation defined in subclause 10.2.3 of IEEE Std 1363a-2004.

Copyright © 2016 IEEE. All rights reserved.

17. ECC Encrypted Control Format

17.1 Overview

The ECC Encrypted Control Format (EECF) is used to transport an existing control AVTPDU to another end station using elliptic curve public key cryptography.

17.2 ECC Encrypted Control AVTPDU

17.2.1 Overview

This encapsulation shall use the alternative header and shall have a **subtype** field of EECF (see 4.4.3.2).

The **h** (header_specific) bit (4.4.3.3) of the common header is used to define the following:

— **r** field: 1 bits

This encapsulation uses the following alternative header field directly (see 4.4.6 for definitions):

— **version** field: 3 bits

This encapsulation uses the **alternative_data_payload** field to define the following fields:

— **enc**: 4 bits
— **reserved**: 5 bits
— **encrypted_payload_length**: 11 bits
— **key_id**: 8 octets
— **encrypted_payload**: variable length, up to and including 1488 octets

The fields are shown in Figure 84.

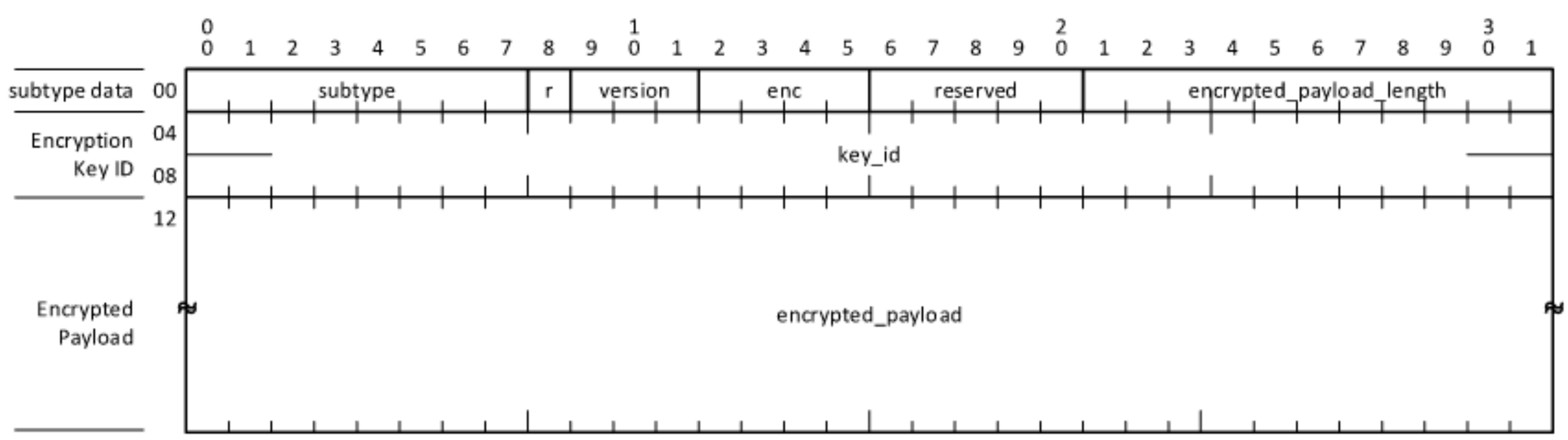


Figure 84—ECC encrypted control AVTPDU

17.2.2 enc field

The **enc** field defines the ECC encryption algorithm variant used by the EECF AVTPDU.

Copyright © 2016 IEEE. All rights reserved.

The value of the **enc** field is defined in Table 48.

Table 48 —enc field values

Value	Name	Description	Reference
0_{16}	ECC1	Elliptic curve variant 1	Subclause 17.3
1 to F_{16}	—	Reserved	—

17.2.3 encrypted_payload_length field

The **encrypted_payload_length** field specifies the length of the **encrypted_payload** field in octets.

17.2.4 key_id field

The 8-octet **key_id** field is an EUI-64 used to identify the elliptic curve private key used to encrypt the plaintext payload and generate the **encrypted_payload** field.

NOTE—The specific mechanism of how EUI-64 values are defined and assigned is outside the scope of this standard. For more information see subclause 7.6.1.1 in IEEE Std 1722.1-2013.

17.2.5 encrypted_payload field

The **encrypted_payload** field contains the output of the ECC encryption algorithm variant as selected by the **enc** field, and is limited in length to 1488 octets.

17.3 Elliptic curve variant 1 algorithm

17.3.1 Overview

When the **enc** field value is ECC1, the **encrypted_payload** field contains the ECC1 plaintext payload (see 17.3.2) encrypted with elliptic curve cryptography as defined in subclause 11.3.2 of IEEE Std 1363a-2004 with the following scheme options:

— The secret value derivation primitive shall be ECSVDP-DHC (subclause 7.2.2 of IEEE Std 1363-2000), without compatibility with ECSVDP-DH

— The method for encrypting the message shall be KDF2 (subclause 13.2 of IEEE Std 1363a-2004), using the hash function SHA-256 (subclause 14.1.3 of IEEE Std 1363a-2004)

— The symmetric encryption scheme shall be AES-CBC-IV0 (subclause 14.3.2 of IEEE Std 1363a-2004)

— ECIES shall be used in DHAES (Diffie-Hellman augmented encryption scheme) mode (subclause D.5.3 of IEEE Std 1363a-2004)

— The elliptic curve primitives shall be “X-coordinate only representation: EC2OSP-X and OS2ECP-X,” as defined in subclause 5.5.6.3 of IEEE Std 1363a-2004

Decryption of the **encrypted_payload** field is achieved by using the algorithm defined in subclause 11.3.3 of IEEE Std 1363a-2004.

Copyright © 2016 IEEE. All rights reserved.

17.3.2 ECC1 plaintext payload

17.3.2.1 Overview

The plaintext payload for the ECC1 elliptic curve algorithm is composed of the following fields:

— **encryption_timestamp**: 8 octets

— **encapsulated_avtpdu**: variable length

The fields are shown in Figure 85.

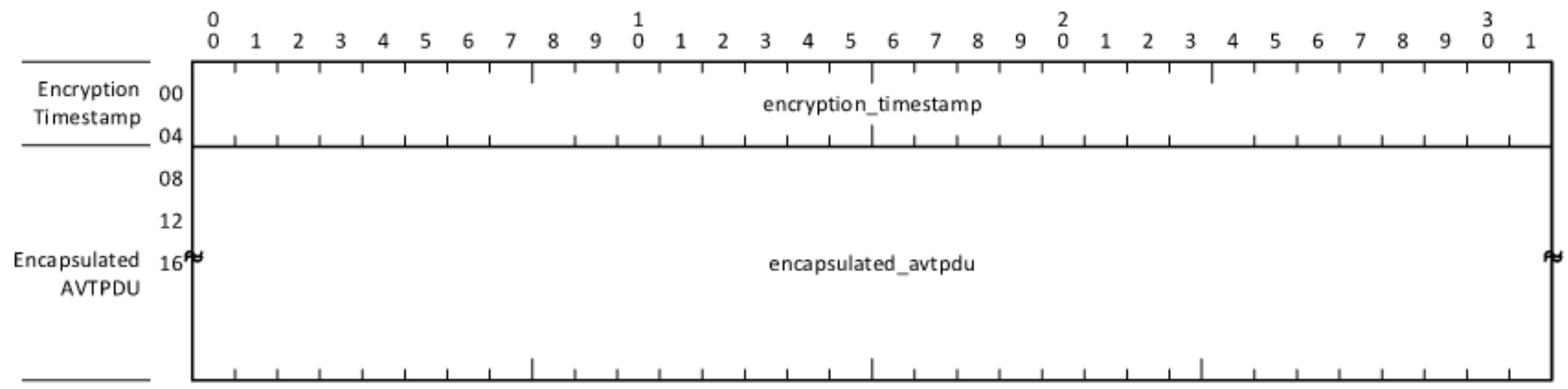


Figure 85—ECC plaintext payload

17.3.2.2 encryption_timestamp field

The **encryption_timestamp** field is an 8-octet field representing gPTP time in nanoseconds at the time that the signing algorithm was applied.

The **encryption_timestamp** is constructed identically to the **CRF_timestamp** as defined in 10.4.13 and calculated in Equation (13).

17.3.3 encapsulated_avtpdu

The **encapsulated_avtpdu** field contains a complete AVTPDU starting at the **subtype** field.

Copyright © 2016 IEEE. All rights reserved.

Annex A

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] 1394 Trade Association, 2004006, AV/C Digital Interface Command Set General Specification Version 4.2.[7]

[B2] AES-2id-2012, AES Information Document for Digital Audio Engineering—Engineering Guidelines for the Use of AES3 Interface.[8]

[B3] AES10-2008, Revision of AES10-2003 AES Recommended Practice for Digital Audio Engineering—Serial Multichannel Audio Digital Interface (MADI).

[B4] AES-10id-2005 (r2011) AES Information Document for Digital Audio Engineering—Engineering Guidelines for the Multichannel Audio Digital Interface AES10 (MADI).

[B5] AES11-2009, AES Recommended Practice for Digital Audio Engineering—Synchronization of Digital Audio Equipment in Studio Operations.

[B6] Digital Content Protection LLC, High-bandwidth Digital Content Protection System—Interface Independent Adaptation.[9]

[B7] Digital Transmission Licensing Administrator (DTLA), Digital Transmission Content Protection (DTCP) Specification Volume 1 (Informational Version).[10]

[B8] Digital Transmission Licensing Administrator (DTLA), DTCP Volume 1, Supplement D, DTCP Use of IEEE 1394 Similar Transports Revision 1.1 (Informational Version).[11]

[B9] IEC 61784-2:2010, Industrial Communication Networks—Profiles—Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3.[12]

[B10] IEC 61966-2-1:1999, Multimedia Systems and Equipment—Colour Measurement and Management—Part 2-1: Colour Management—Default RGB Colour Space—SRGB.

[B11] IEEE Std 802™, IEEE Standard for Local and Metropolitan Area Networks—Overview and Architecture.[13, 14]

[B12] IEEE Std 802.1BA™-2011, Standard for Local and Metropolitan Area Networks—Audio Video Bridging (AVB) Systems.

[B13] IEEE Std 1394.1™-2004, IEEE Standard for High Performance Serial Bus Bridges.

---

[7]Available at http://www.1394ta.org/developers/Specifications.html.
[8]AES standards are available at http://www.aes.org/standards.
[9]Available at http://www.digital-cp.com/sites/default/files/HDCP%20Interface%20Independent%20Adaptation%20Specification%20Rev2_0_0.pdf.
[10]Available at http://www.dtcp.com/documents/dtcp/info-20130605-dtcp-v1-rev-1-7-ed2.pdf.
[11]Available at http://www.dtcp.com/documents/dtcp/info-20070615-dtcp-v1sd-rev-1-p-1.pdf.
[12]IEC publications are available from the Sales Department of the International Electrotechnical Commission, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iec.ch/). IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.
[13]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[14]The IEEE standards or products referred to in this clause are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.

Copyright © 2016 IEEE. All rights reserved.

[B14] ISO 11664-1:2007/CIE S 014-1/E:2006, Colorimetry—Part 1: CIE Standard Colorimetric Observers.[15]

[B15] ISO 10681-1:2010, Communication on FlexRay—Part 1: General Information and Use Case Definition.

[B16] ISO 10681-1:2010, Communication on FlexRay—Part 2: Communication Layer Services.

[B17] ISO 11898, Controller Area Network (CAN).

[B18] ISO 17987, Road vehicles—Local Interconnect Network.

[B19] ITU-R Recommendation BT.601-7 (03/2011), Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios.

[B20] ITU-R Recommendation BT.656, Interface for Digital Component Video Signals in 525-Line and 625-Line Television Systems Operating at the 4:2:2 Level of Recommendation ITU-R BT.601.

[B21] ITU-R Recommendation BT.709-6 (06/2015), Parameter Values for the HDTV Standards for Production and International Programme Exchange.[16]

[B22] ITU-R Recommendation BT.2020-1 (06/2014), Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange.

[B23] ITU-T Recommendation H.264 (02/2014), Advanced Video Coding for Generic Audiovisual Services.

[B24] MIDI Manufacturers Association (MMA), MMA Payload Format Specification for AVTP.[17]

[B25] MOST Specification Rev. 3.0, Errata 2, 07/2010.[18]

[B26] NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.[19]

[B27] SMPTE ST 291-1:2011, Ancillary Data Packet and Space Formatting.

[B28] SMPTE ST 372:2011, Dual Link 1.5 Gb/s Digital Interface for 1920 × 1080 and 2048 × 1080 Picture Formats.[20]

[B29] SMPTE ST 424M:2006, 3 Gb/s Signal/Data Serial Interface.

[B30] SMPTE ST 2035:2009, Am1:2012, Audio Channel Assignments for Digital Television Recorders (DTRs)—Amendment 1.

[B31] SMPTE RP 168:2009, Definition of Vertical Interval Switching Point for Synchronous Video Switching.

[15]ISO publications are available at http://www.iso.org.
[16]Available at https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-E.pdf.
[17]The MIDI Manufacturers Association Inc., La Habra, CA, USA http://www.midi.org/.
[18]Available at http://www.mostcooperation.com/publications/specifications-organizational-procedures/index.html.
[19]Available at http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.
[20]SMPTE publications are available at http://standards.smpte.org.

Copyright © 2016 IEEE. All rights reserved.

Annex B

(normative)

MAC Address Acquisition Protocol

B.1 Overview

The media access control (MAC) Address Acquisition Protocol (MAAP) is designed to provide a way to allocate dynamically the multicast MAC addresses needed by AVTP.

A block of multicast MAC addresses has been reserved for the use of AVTP. These addresses are listed in Table B.9.

The MAAP specifies a mechanism to allocate multicast MAC addresses dynamically in a specified address range.

Any application that uses addresses from the MAAP dynamic allocation pool shall implement the MAAP and MAAP shall be used to allocate these addresses.

MAAP can be used to request a single address or a range of consecutive addresses.

The MAAP uses a probe, announce, and defend mechanism in a peer-to-peer configuration to acquire addresses to be used by AVTP. An overview of the MAAP is described in this clause, and the detailed MAAP is described in B.3.

To acquire an address range for use, an implementation of MAAP executes the following steps:

— Select an address range from the MAAP dynamic allocation pool.
— Send a series of MAAP_PROBE protocol data units (PDUs) to determine whether the address range is already in use.
— Listen for MAAP_DEFEND PDUs indicating the address range is in use.
— Repeat the above steps until an unused address range has been found.

Once an address range has been acquired, MAAP announces and defends the address range as follows:

a) Send MAAP_ANNOUNCE PDUs periodically to inform the network of address ranges that are currently in use. This allows networks that are joined after the MAAP has been run to identify conflicts.

b) Listen for MAAP_PROBE PDUs and send MAAP_DEFEND PDUs in response to any address ranges that conflict with previously acquired address range(s).

c) Listen for MAAP_ANNOUNCE PDUs that conflict with previously acquired address range(s).

d) If a conflicting MAAP_ANNOUNCE PDU is received, then MAAP discontinues use of the conflicting address range(s) and acquires new address range(s).

Copyright © 2016 IEEE. All rights reserved.

B.2 Protocol Message Format

B.2.1 Overview

This encapsulation uses the common control header and has a **subtype** field of MAAP (see 4.4.3.2).

This encapsulation uses the following common control fields (see 4.4.3 and 4.4.5 for definitions):

— **sv**: 1 bit

— **version**: 3 bits

— **control_data_length**: 11 bits

Specific use is given for the following common control field:

— **stream_id**: 8 octets

This encapsulation uses the **format_specific_data** and the **control_data_payload** fields of the common control header to define the following format-specific fields:

— message_type: 4 bits

— maap_version: 5 bits

— requested_start_address: 6 octets

— requested_count: 2 octets

— conflict_start_address: 6 octets

— conflict_count: 2 octets

These fields are shown in Figure B.1.

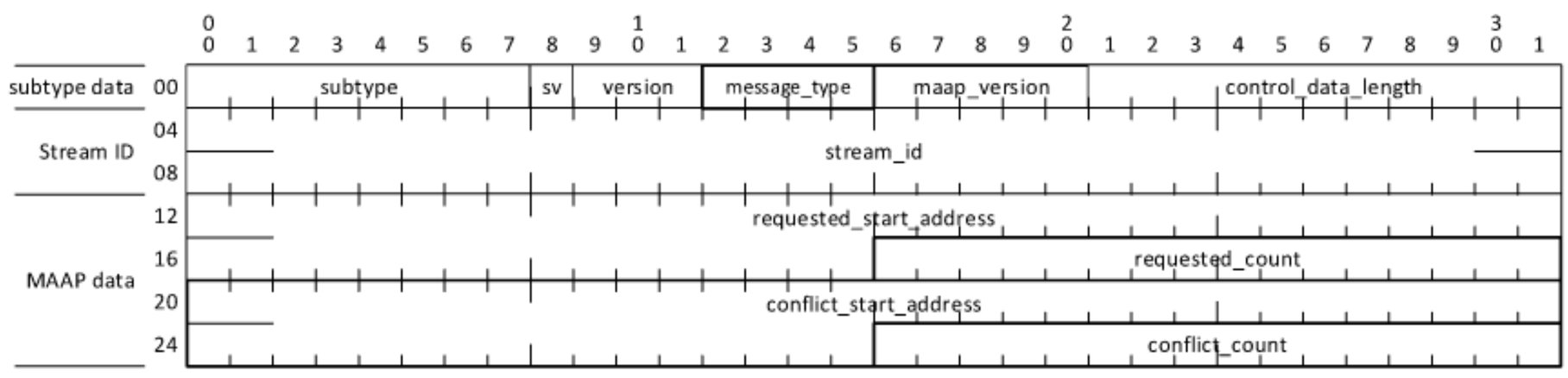


Figure B.1—MAAP PDU Format

All MAAP_PROBE and MAAP_ANNOUNCE frames are sent with a multicast destination MAC address set to the reserved MAAP multicast address defined in Table B.10.

All MAAP_DEFEND frames are sent with a destination MAC address set to the source MAC address received in the MAAP_PROBE frame that caused the sending of the MAAP_DEFEND frame.

The source MAC address shall be set to the MAC address of the sender.

Copyright © 2016 IEEE. All rights reserved.

The EtherType field shall be the AVTP EtherType defined in 4.4.2.

The **control_data_length** field shall be set to 16_{10} in all MAAP frames.

B.2.2 message_type field

The **message_type** field contains one of the defined MAAP message types as defined in Table B.1. If a MAAP Audio Video Transport Protocol Data Unit (AVTPDU) is received with a reserved **message_type**, the MAAP AVTPDU shall be ignored.

Table B.1—MAAP message types

Value	Function	Description
0_{16}	—	Reserved
1_{16}	MAAP_PROBE	Probe MAC address(es) PDU
2_{16}	MAAP_DEFEND	Defend address(es) response PDU
3_{16}	MAAP_ANNOUNCE	Announce MAC address(es) acquired PDU
4 to F_{16}	—	Reserved

B.2.3 maap_version field

B.2.3.1 Overview

The **maap_version** field identifies the version of MAAP being used. The current version of MAAP is one (1).

To be compatible with current protocol versions, while allowing the protocol to be extended in the future, the requirements listed in B.2.3.2 through B.2.3.4 shall be met by the implementation.

B.2.3.2 MAAP AVTPDUs that carry a version higher than the receiver version

MAAP AVTPDUs that carry a protocol version higher than the protocol version implemented by the receiver shall be interpreted according to the protocol definition of the receiver's implemented version.

All MAAP AVTPDUs received that contain a higher version number and a message type that is defined in the implemented version of MAAP shall be interpreted using the implemented version of MAAP, ignoring all unknown fields. This requires that future versions of MAAP maintain compatibility with the message types implemented in all previous versions of MAAP.

All MAAP AVTPDUs received that contain a higher version number and a message type that is not defined in the implemented version of MAAP shall be ignored.

B.2.3.3 MAAP AVTPDUs that carry a version equal to the receiver version

MAAP AVTPDUs that carry a protocol version equal to the protocol version implemented by the receiver shall be interpreted according to the protocol definition of the receiver's implemented version.

Copyright © 2016 IEEE. All rights reserved.

B.2.3.4 MAAP AVTPDUs that carry a version lower than the receiver version

MAAP AVTPDUs that carry a protocol version lower than the protocol version implemented by the receiver shall be interpreted according to the protocol definition corresponding to the protocol version received in the MAAP AVTPDU. This requires that future versions of MAAP maintain the ability to interpret MAAP AVTPDUs from all previous versions of MAAP.

B.2.4 stream_id field

The **stream_id** field is not used in MAAP and shall be set to zero (0).

B.2.5 requested_start_address field

The **requested_start_address** in a MAAP_PROBE or MAAP_ANNOUNCE message is the first address of a consecutive range of addresses being requested.

In a MAAP_DEFEND message, this field is set to the **requested_start_address** value received in the MAAP_PROBE or MAAP_ANNOUNCE PDU that initiated the transaction.

B.2.6 requested_count field

The **requested_count** field in a MAAP_PROBE or MAAP_ANNOUNCE message is the number of addresses being requested. If only a single address is being requested, this field is set to one (1).

In a MAAP_DEFEND message, this field is set to the **requested_count** value received in the MAAP_PROBE or MAAP_ANNOUNCE PDU that initiated the transaction.

B.2.7 conflict_start_address field

The **conflict_start_address** field in a MAAP_DEFEND message is set to the first address that conflicts with a requested address range from a MAAP_PROBE PDU.

In all other MAAP messages, this field is set to zero (0).

B.2.8 conflict_count field

The **conflict_count** field in a MAAP_DEFEND message is set to the number of addresses in the allocated address range, beginning with **conflict_start_address**, that conflicts with a requested address range from a MAAP PROBE PDU.

In all other MAAP messages, this field is set to zero (0).

Copyright © 2016 IEEE. All rights reserved.

B.3 Protocol specification

B.3.1 Overview

The conventions shown in Table B.2 are used in the abbreviations used in this subclause.

Table B.2—MAAP notational conventions

Convention	Description
rXXX	Receive PDU XXX
sXXX	Send PDU XXX
XXX!	State machine event

The MAAP is defined by the MAAP state machine (Table B.7). The following abbreviations are used in the state machine descriptions. For their meaning, see B.3.3, B.3.5, and B.3.6.

Table B.3—Protocol events

Event	Description
Begin!	Initialize state machine (B.3.5.1)
Release!	Release an address range (B.3.5.2)
Restart!	Restart the state machine (B.3.5.3)
ReserveAddress!	Reserve an address (B.3.5.4)
rProbe!	Receive a MAAP PROBE PDU (B.3.5.5)
rDefend!	Receive a MAAP DEFEND PDU (B.3.5.6)
rAnnounce!	Receive a MAAP ANNOUNCE PDU (B.3.5.7)
probeCount!	maap_probe_count has decremented to zero (0) (B.3.5.8)
announceTimer!	announce_timer has expired (B.3.4.1)
probeTimer!	probe_timer has expired (B.3.4.2)
PortOperational!	The port transitions to an operational state (B.3.5.9).

Table B.4—Protocol counters

Counter	Description
maap_probe_count	Counter that decrements from MAAP_PROBE_RETRANSMITS to zero (0)

Table B.5—Protocol actions

Action	Description
generate_address	Generate a random address range (B.3.6.1)
init_maap_probe_count	Initialize maap_probe_count (B.3.6.2)
dec_maap_probe_count	Decrement maap_probe_count (B.3.6.3)
compare_MAC	Compare the received and local MAC address (B.3.6.4)
probe_timer	Probe period timer (B.3.4.2)
announce_timer	Announce period timer (B.3.4.1)
sProbe	Send a MAAP_PROBE PDU (B.3.6.5)
sDefend	Send a MAAP_DEFEND PDU (B.3.6.6)
sAnnounce	Send a MAAP_ANNOUNCE PDU (B.3.6.7)
-x-	Inapplicable event/state combination. No action or state.

Copyright © 2016 IEEE. All rights reserved.

Timers are used in the state machine descriptions to cause actions to be taken after defined time periods have elapsed. The following terminology is used in the state machine descriptions to define timer states and actions that can be performed upon them:

a) A timer is said to be running if the most recent action performed upon it was a start.

b) A running timer is said to have expired when the timer period associated with the timer has elapsed since the most recent start action took place.

c) A timer is said to be stopped if it has expired or if the most recent action performed upon it was a stop action.

d) A start action sets a stopped timer to the running state and associates a time period with the timer. This time period supersedes any periods that might have been associated with the timer by previous state events.

e) A stop action sets a timer to the stopped state.

Table B.6—Protocol states

State	Description
INITIAL	Initialize the state machine
PROBE	Probe the availability of an address range
DEFEND	Defend an acquired address range

B.3.2 State machine

A participant in MAAP maintains a single instance of this state machine for each address range currently allocated, or that is in the process of being allocated. The state machine defines events, actions, and state changes to be executed in response to a received packet or event. Functions, events, and state changes in each state table entry shall be executed sequentially. State changes are indicated by the new state name listed in the state table. If no state name is listed in an entry, then no state change occurs and the machine remains in its current state. State table entries marked with an "-x-" indicate that no functions, events, or state changes occur and that the state machine remains in its current state. An entry that contains a state change/event function indicates that the state change is to occur first and then is immediately followed by the execution of the event in the destination state.

The detailed operation of this state machine is described in Table B.7.

Copyright © 2016 IEEE. All rights reserved.

Table B.7—MAAP state machine

		STATE		
		INITIAL	**PROBE**	**DEFEND**
EVENT	**Begin!**	generate_address[a] ReserveAddress!	-x-	-x-
	Release![c]	-x-	Stop probe_timer INITIAL	Stop announce_timer INITIAL
	Restart!	generate_address ReserveAddress!	-x-	-x-
	ReserveAddress!	init_maap_probe_count Start probe_timer sProbe PROBE	-x-	-x-
	rProbe![b]	-x-	compare_MAC[d] Stop probe_timer INITIAL/Restart!	sDefend
	rDefend![b]	-x-	Stop probe_timer INITIAL/Restart!	compare_MAC[d] Stop announce_timer INITIAL/Restart!
	rAnnounce![b]	-x-	Stop probe_timer INITIAL/Restart!	compare_MAC[d] Stop announce_timer INITIAL/Restart!
	probeCount!	-x-	Stop probe_timer Start announce_timer sAnnounce DEFEND	-x-
	announcetimer!	-x-	-x-	Start announce_timer sAnnounce
	probetimer!	-x-	Start probe_timer sProbe dec_maap_probe_count	-x-
	PortOperational!	generate_address[a] ReserveAddress!	Stop probe_timer INITIAL/Restart!	Stop announce_timer INITIAL/Restart!

[a] A Begin! or PortOperational! event can be initiated with an assigned address range or MAAP can select an address range with the generate_address function. If an address range is supplied with the Begin! or PortOperational! event, generate_address will not be called and the supplied address range will be used. If the application has previously obtained an address range and has access to persistent storage, the application may record the previous address range and attempt to reuse the saved address range.

[b] Only received MAAP_PROBE, MAAP_DEFEND, and MAAP_ANNOUNCE PDUs that conflict with the address range associated with this state machine generate rProbe!, rDefend!, and rAnnounce! events. All MAAP_PROBE, MAAP_DEFEND, and MAAP_ANNOUNCE PDUs that do not conflict with the address range associated with this state machine are ignored.

[c] After a Release! event is received and the state machine has returned to INITIAL state, the address range associated with this state machine is considered to be free, and the state machine can be destroyed.

[d] If the compare_MAC function returns TRUE then no further processing or protocol action is taken and the protocol state does not change.

B.3.3 MAAP constants

Constant values used by the MAAP are listed in Table B.8.

Copyright © 2016 IEEE. All rights reserved.

Table B.8—MAAP probe constant values

Constant	Value
MAAP_PROBE_RETRANSMITS	3
MAAP_PROBE_INTERVAL_BASE	500 ms
MAAP_PROBE_INTERVAL_VARIATION	100 ms
MAAP_ANNOUNCE_INTERVAL_BASE	30 s
MAAP_ANNOUNCE_INTERVAL_VARIATION	2 s

B.3.4 Protocol timers

B.3.4.1 announce_timer

The announce period timer, announce_timer, controls the period of time that separates the transmission of MAAP_ANNOUNCE PDUs. The announce period timer is set to a random value T, in the range MAAP_ANNOUNCE_INTERVAL_BASE < T < MAAP_ANNOUNCE_INTERVAL_BASE + MAAP_ANNOUNCE_INTERVAL_VARIATION.

NOTE—A random variation is added to the Announce and Probe period times to avoid synchronized bursts of frames flooding the network.

B.3.4.2 probe_timer

The probe period timer, probe_timer, controls the period of time that separates the transmission of MAAP_PROBE PDUs. The probe period timer is set to a random value T, in the range MAAP_PROBE_INTERVAL_BASE < T < MAAP_PROBE_INTERVAL_BASE + MAAP_PROBE_INTERVAL_VARIATION.

B.3.5 Protocol events

B.3.5.1 Begin!

The state machine is initialized or reinitialized.

B.3.5.2 Release!

A Release! event signals that the address range associated with this instance of the state machine is no longer in use and is no longer to be defended.

B.3.5.3 Restart!

A Restart! event signals that an address range conflict has been detected and that the state machine will be restarted with a new address range.

Copyright © 2016 IEEE. All rights reserved.

B.3.5.4 ReserveAddress!

A ReserveAddress! event signals that an address range is available to start the process of sending MAAP_PROBE PDUs to acquire the address range for use.

B.3.5.5 rProbe!

An rProbe! event signals that a MAAP_PROBE PDU has been received that contains an address range that conflicts with the address range assigned to this state machine.

B.3.5.6 rDefend!

An rDefend! event signals that a MAAP_DEFEND PDU has been received that contains an address range that conflicts with the address range assigned to this state machine.

B.3.5.7 rAnnounce!

An rAnnounce! event signals that a MAAP_ANNOUNCE PDU has been received that contains an address range that conflicts with the address range assigned to this state machine.

B.3.5.8 probeCount!

A probeCount! event signals that the dec_maap_probe_count has decremented the maap_probe_count to a value less than or equal to zero (0).

B.3.5.9 PortOperational!

A PortOperational! event signals that the port has entered an operational state. A port is in an operational state when it is able to transmit and receive AVTPDUs.

B.3.6 Protocol actions

B.3.6.1 generate_address

The generate_address function randomly selects an address range based on the number of addresses requested by the application. The address range is selected from the MAAP dynamic allocation pool (Table B.9). The generate_address function selects the address range using a pseudo-random number generator with a uniform distribution across the MAAP dynamic allocation pool range.

The pseudo-random number generation algorithm should be chosen so that different hosts do not generate the same sequence of numbers for subsequent calls to generate_address. The pseudo-random number generation algorithm should have a maximum length sequence of no less than $2^{32} - 1$. The pseudo-random number generator shall be seeded using the least-significant (most rapidly varying) octets of the sum of the requestor's IEEE 802 MAC address and the local real-time clock. The use of a ones-complement (end-around carry) sum is suggested but not required.

Copyright © 2016 IEEE. All rights reserved.

NOTE—Information about previously allocated address ranges gleaned from received MAAP_ANNOUNCE messages can be used to improve the performance of the generate_address function to avoid selection of addresses that are already in use. The specific algorithms to do this are out of scope of this standard.

B.3.6.2 init_maap_probe_count

The init_maap_probe_count function sets the value of maap_probe_count to MAAP_PROBE_RETRANSMITS.

B.3.6.3 dec_maap_probe_count

The dec_maap_probe_count function decrements the value of maap_probe_count by one (1) and sends a probeCount! event if the resulting value of maap_probe_count is less than or equal to zero (0).

B.3.6.4 compare_MAC

The compare_MAC function compares the MAC address from the received MAAP AVTPDU and the receiving station's MAC address. The MAC address comparison shall be performed in octet-wise reverse order; i.e., for the purpose of the unsigned comparison, the least-significant octet is treated as if it were most significant, the most-significant octet is treated as if it were least significant, and the interior octets are compared in reverse order of their normal significance. The compare_MAC function shall return TRUE if the MAC address of the receiving station is numerically lower than the MAC address from the received MAAP AVTPDU.

When the compare_MAC function returns TRUE, then no further processing or protocol action is taken and the protocol state does not change.

NOTE—An octet-wise reverse order comparison is used in the compare_MAC function to make the company-assigned MA-L (OUI), MA-M, or MA-S be the least-significant component of the comparison.

B.3.6.5 sProbe

The sProbe function sends a MAAP_PROBE PDU where:

— message_type field is set to MAAP_PROBE.

— requested_start_address field is set to the start of the address range being requested.

— requested_count field is set to the number of contiguous MAC addresses being requested.

— conflict_start_address field is set to 00:00:00:00:00:00.

— conflict_count field is set to zero (0).

B.3.6.6 sDefend

The sDefend function sends a MAAP_DEFEND PDU where:

— message_type field is set to MAAP_DEFEND.

— requested_start_address field is set to the value of the requested_start_address field from the MAAP_PROBE that caused the sDefend function to be called.

Copyright © 2016 IEEE. All rights reserved.

— requested_count field is set to the value of the requested_count field from the MAAP_PROBE that caused the sDefend function to be called.

— conflict_start_address field is set to the first allocated address that conflicts with the requested address range.

— conflict_count field is set to the count of allocated addresses beginning with conflict_start_address that conflict with the requested address range.

NOTE—A MAAP_PROBE PDU can contain an address range that conflicts with multiple contiguous or noncontiguous address ranges that are being defended. Because an instance of the state machine is maintained for each acquired address range, a separate MAAP_DEFEND PDU will be sent for each address range that conflicts with a received MAAP_PROBE.

B.3.6.7 sAnnounce

The sAnnounce function sends a MAAP_ANNOUNCE PDU where:

— message_type field is set to MAAP_ANNOUNCE.

— requested_start_address field is set to the start of the address range acquired.

— requested_count field is set to the number of addresses acquired.

— conflict_start_address field is set to 00:00:00:00:00:00.

— conflict_count field is set to zero (0).

B.4 Reserved MAAP MAC addresses

There are 256 MAAP addresses that are reserved for the purposes of this standard. The MAAP reserved addresses are defined in Table B.9. MAAP reserved multicast addresses are divided into three categories, including a dynamic allocation pool, a locally administered static allocation pool, and a pool reserved for use by this standard. Addresses other than those ranges defined in Table B.10 may be used, however these addresses must be owned by the implementor or be from a public dynamically assignable pool of addresses. The multicast address pools are defined in Table B.9. Usage of the MAAP reserved address pool is defined in Table B.10.

The address ranges given in Table B.9 are a subset of the Global MA-L usage table defined in Annex D.

Table B.9—MAAP multicast addresses

Address range	Function	Description
91:E0:F0:00:00:00 to 91:E0:F0:00:FD:FF	MAAP dynamic allocation pool	These addresses are available for dynamic allocation by the MAAP
91:E0:F0:00:FE:00 to 91:E0:F0:00:FE:FF	MAAP locally administered pool	These addresses are reserved to be statically allocated
91:E0:F0:00:FF:00 to 91:E0:F0:00:FF:FF	MAAP reserved pool	Use of these addresses is defined in Table B.10

Table B.10—Reserved multicast MAAP MAC addresses

Address	Function	Description
91:E0:F0:00:FF:00	MAAP	Multicast address used for MAAP frames
91:E0:F0:00:FF:01 to 91:E0:F0:00:FF:FF	—	Reserved

Copyright © 2016 IEEE. All rights reserved.

Annex C

(informative)

IEEE 802.3 Media-specific encapsulation

C.1 Introduction

This annex for end stations describes the IEEE Std 802.3 encapsulation of Audio/Video Transport Protocol Data Units (AVTPDUs). The purpose of this annex is to aid in the understanding of the relationship of Audio/Video Transport Protocol (AVTP) to the overall Ethernet protocol stack. Only details of the encapsulation for IEEE Std 802.3 are included in this annex.

The IEEE 802.1 Time-Sensitive Networking (TSN) standards require that reserved data frames (as opposed to legacy data frames) be marked with a 3-bit priority code point (PCP) so bridges can detect these frames as SR class A or SR class B and treat these frames differently from non-TSN reserved frames. Without this marking of reserved frames, the assurances and performance of TSN cannot be met. This marking of reserved frames is not the only requirement for a successful TSN-based network (see the full set of IEEE 802.1 TSN standards), but it is a necessary one, so its proper usage from an end station point of view is discussed here.

This annex covers the following fields:

a) Destination media access control (MAC) address: 6 octets

b) Source MAC address: 6 octets

c) IEEE Std 802.1Q protocol header: 4 octets consisting of:

   1) Tagged protocol identifier (TPID): 2 octets

   2) Drop eligible indicator (DEI): 1 bit

   3) PCP: 3 bits

   4) Virtual Local Area Network (VLAN) identifier: 12 bits

C.2 IEEE Std 802.3 encapsulation

C.2.1 Stream Reservation Protocol (SRP) reserved frames

C.2.1.1 Overview

AVTP streams that use stream-reservation capabilities such as those provided by SRP use an IEEE Std 802.1Q (VLAN-tagged) header.

For these reserved streams, the EtherType field immediately following the source MAC address is known as the TPID field and is set to 8100_{16}. For this case, the AVTP EtherType is at an offset 4 octets past the start of this field.

Figure C.1 shows an AVTP frame encapsulated within an IEEE Std 802.3 frame with an IEEE Std 802.1Q header (also known as an IEEE Std 802.1Q VLAN Tag field).

Copyright © 2016 IEEE. All rights reserved.

NOTE—The AVTP frame's common header and data, as discussed in 4.4.3, immediately follows the AVTP frame's EtherType field.

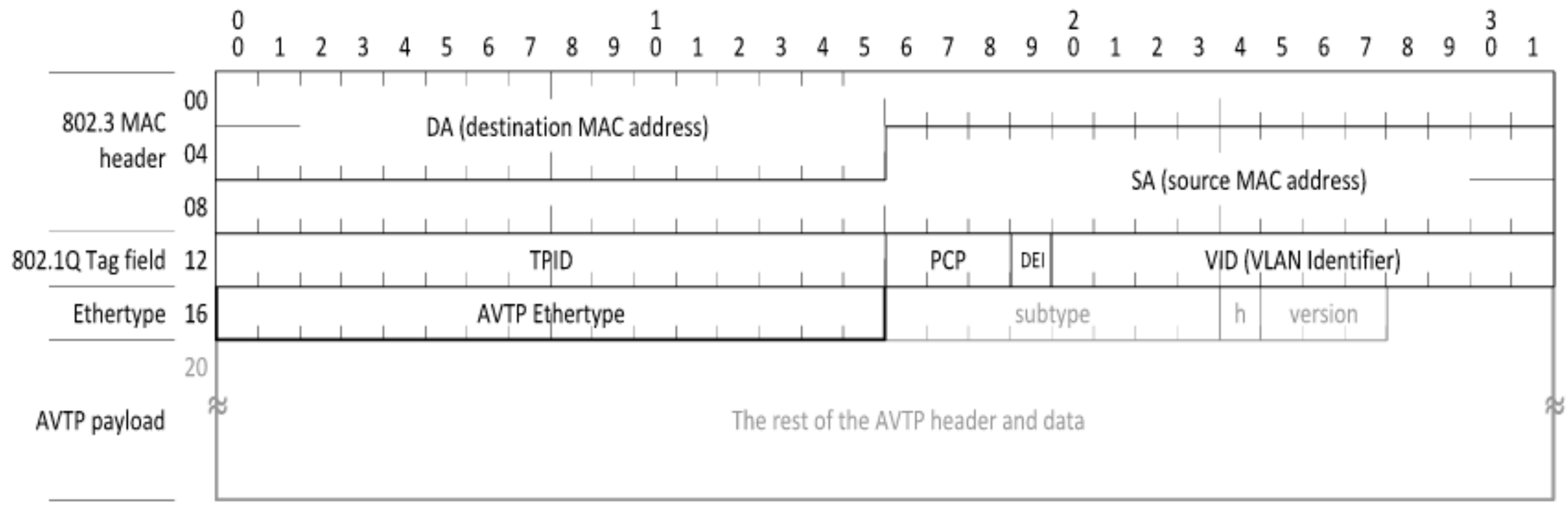


Figure C.1. AVTPDU in an IEEE Std 802.3 frame with an IEEE Std 802.1Q Tag field

C.2.1.2 DA (IEEE Std 802.3 destination MAC address) field

The MAC destination address of each stream will be unique for the layer 2 network and may either be a locally administered unicast address or a multicast address (as defined in SRP). Multicast addresses may be assigned by use of MAAP defined in Annex B. The use of MAAP allows for multicast addresses to be unique per AVTP stream.

C.2.1.3 SA (IEEE Std 802.3 source MAC address) field

The MAC source address contains the sender's unicast MAC address.

C.2.1.4 TPID (IEEE Std 802.1Q–tagged protocol identifier) field

The TPID field is set to a value of 8100_{16} as required by IEEE Std 802.1Q.

C.2.1.5 PCP (priority code point) field

For all AVTP frames that are part of an SRP reserved stream, AVTP Talkers set the PCP value to the SRclassPriority (SRP, 35.2.2.9.3), which it learns via the MSRP Domain attribute for either stream SR class A traffic or stream SR class B traffic.

All AVTP Listeners are able to receive and process stream data frames with any PCP value (the SR class A and SR class B defaults are used unless modified by management).

C.2.1.6 DEI field

Subclause 9.6 of IEEE Std 802.1Q-2014 defines usage of the DEI (drop_eligible) bit. In most SRP applications this bit will be zero (0).

Copyright © 2016 IEEE. All rights reserved.

C.2.1.7 VID (VLAN ID) field

The VLAN identifier field is used to indicate the IEEE Std 802.1Q VLAN on which an AVTP frame is to be processed. It is not to be used for any other purpose (i.e., it is not to be used as a stream identifier in any way). VLANs are generally set up and managed in bridges. End stations (Talkers and Listeners in this standard) may or may not be VLAN aware. A VLAN-aware station is one that can process information by management that tells the device what VID or VIDs it is a member of and what VID each stream is to use. Special care is needed when using VLAN aware stations, and all VLAN-aware stations default to a mode where they act as if they are not VLAN aware until configured otherwise by management.

If an AVTP Talker is not VLAN aware, it transmits all stream data frames with a VID of SRclassVID (SRP, 35.2.2.9.4), which it learns via the MSRP Domain attribute. If a Talker is directly connected to a Listener, it may use a VID of SR_PVID (SRP, 35.2.1.4).

If an AVTP Talker is VLAN aware, it transmits stream data frames with the VID VLAN management has assigned to it.

To receive stream data frames, all AVTP Listeners request membership in the Talker's VLAN. A Listener does not need to be VLAN aware to do this; it simply needs to issue an MVRP membership request for the VID contained in the Talker Advertise.

Conceptually, all Talker functions and all Listener functions are not VLAN aware. If a Talker device needs to be VLAN aware, it can be thought of as a non–VLAN-aware Talker function that always transmits streams with a VID of SR_PVID followed by a logical two-port bridge function that assigns a VID to streams from the particular Talker function. A device with two Talker functions that needs to transmit separate streams out with separate VIDs can be thought of as two non–VLAN-aware Talker functions connected with a three-port bridge function that assigns separate VIDs for each Talker. Likewise, if a Listener device requires VLAN filtering, it can be seen as a non–VLAN-aware Listener that ignores all VID processing with a two-port bridge function in front of it that performs VLAN filtering.

C.2.1.8 AVTP EtherType field

This field contains the AVTP-assigned EtherType, as defined in 4.4.2.

C.2.2 Non-reserved, VLAN-tagged frames

C.2.2.1 Overview

AVTP frames that do not use stream-reservation capabilities such as those provided by SRP, but use an IEEE Std 802.1Q (VLAN-tagged) header, use a packet encapsulation as described in this subclause.

The EtherType field immediately following the source MAC address is known as the TPID field and is set to 8100_{16}. For this case, the AVTP EtherType is at an offset 4 octets past the start of this field.

Figure C.1 shows an AVTP frame encapsulated within an IEEE Std 802.3 frame with an IEEE Std 802.1Q header (also known as an IEEE Std 802.1Q VLAN Tag field).

NOTE—The AVTP frame's common header and data, as discussed in 4.4.3, immediately follows the AVTP frame's EtherType field.

Copyright © 2016 IEEE. All rights reserved.

C.2.2.2 DA (IEEE Std 802.3 destination MAC address) field

The MAC destination address of each frame may be unicast, multicast, or broadcast depending on the specification of the usage of each AVTP frame.

C.2.2.3 SA (IEEE Std 802.3 source MAC address) field

The MAC source address contains the sender's unicast MAC address.

C.2.2.4 TPID (IEEE Std 802.1Q tagged protocol identifier) field

The TPID field is set to a value of 8100_{16} as required by IEEE Std 802.1Q.

C.2.2.5 PCP (priority code point) field

AVTP Talkers set the PCP value to a value other than the FQTSS-specified default or management-specified value for either SR class A traffic or SR class B traffic.

All AVTP Listeners are able to receive and process stream data frames with any PCP value (the SR class A and SR class B defaults are used unless modified by management).

C.2.2.6 DEI field

Subclause 9.6 of IEEE Std 802.1Q-2014 defines usage of the DEI (drop_eligible) bit.

C.2.2.7 VID (VLAN ID) field

The VID field is set to the value assigned by management and is outside the scope of this standard.

C.2.2.8 AVTP EtherType field

This field contains the AVTP-assigned EtherType, as defined in 4.4.2.

C.2.3 Non-reserved, untagged frames

C.2.3.1 Overview

Figure C.2 shows an AVTP frame encapsulated within an IEEE Std 802.3 frame without an IEEE Std 802.1Q header. This format is used for AVTP frames (such as some control formats) that do not use stream-reservation capabilities such as those provided by SRP.

NOTE—The AVTP frame's common header and data, as discussed in 4.4.3, immediately follows the AVTP frame's EtherType field.

Copyright © 2016 IEEE. All rights reserved.

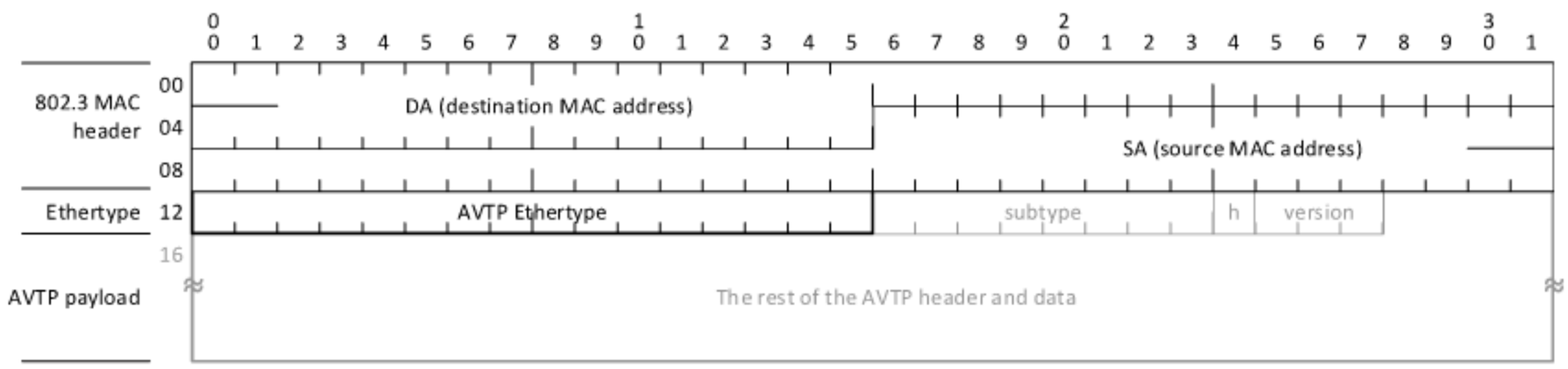


Figure C.2. AVTPDU in an IEEE Std 802.3 frame without an IEEE Std 802.1Q Tag field

C.2.3.2 DA (IEEE Std 802.3 destination MAC address) field

The MAC destination address of each frame may be unicast, multicast, or broadcast depending on the specification of the usage of each AVTP control frame.

C.2.3.3 SA (IEEE Std 802.3 source MAC address) field

The MAC source address contains the sender's unicast MAC address.

C.2.3.4 AVTP Ethertype field

This field contains the AVTP-assigned EtherType, as defined in 4.4.2.

C.3 Other considerations

TSN requires that some IEEE Std 802.3–compliant device modes of operation cannot be used. These requirements are documented in IEEE Std 802.1BA™-2011 [B12]. For informative purposes, some of the TSN rules are listed as follows:

a) Link speed is 100 Mbit/s or faster.

b) Link duplex is full duplex.

c) MAC Pause is not used on TSN links.

d) Priority Pause is not used on TSN priorities.

e) The maximum size of the MAC Client data portion of a frame never exceeds 1500 octets.

Copyright © 2016 IEEE. All rights reserved.

Annex D

(normative)

MA-L usage

Audio/Video Transport Protocol (AVTP) has been assigned MA-L 90:E0:F0. Table D.1 outlines the allocation of the addresses within this range.

Table D.1—AVTP multicast media access control (MAC) address allocation

Address range	Function	Description
91:E0:F0:00:00:00 to 91:E0:F0:00:FF:FF	MAAP	See Annex B (specifically B.4)
91:E0:F0:01:00:00 to 91:E0:F0:01:FF:FF	IEEE Std 1722.1	Reserved for use by IEEE Std 1722.1
91:E0:F0:02:00:00 to 91:E0:F0:FF:FF:FF	Reserved	Reserved

NOTE—Ethernet uses the least-significant bit of the first octet of a MAC address to designate multicast addresses. AVTP has been assigned the entire 90:E0:F0 range, but only the addresses in the multicast range (91:E0:F0:00:00:00 through 91:E0:F0:FF:FF:FF are shown in Table D.1.

Copyright © 2016 IEEE. All rights reserved.

Annex E

(informative)

Diagnostics and counters

E.1 Audio/Video Transport Protocol (AVTP) stream counters

This clause defines a set of counters that are maintained on a per-stream basis. No conclusions about product performance between two vendors can be can be drawn by comparing these numbers. The purpose of these counters is to ascertain the relative health of a particular device. All counters are implemented as unsigned 32-bit integers. Each counter is incremented on a per-event basis. Each counter has a maximum value of $2^{32} - 1$ and rolls over to zero (0) the next time it is incremented.

Counters are valid from the creation of the stream by completing a Stream Reservation Protocol (SRP) reservation, which is established when SRP Talker Advertise (TA) and Listener Ready (LR) messages have been exchanged. All counters are reset to zero on the creation of the SRP reservation at the Talker or Listener. The completion of the SRP Reservation occurs at different times at the Talker and the possible multiple Listeners. Therefore counters that exist on both the Talker and Listener such as FRAMES_TX on the Talker and FRAMES_RX on the Listener are not expected to contain identical values.

Table E.1—AVTP per-stream counters

Name	Description	Talker/Listener
MEDIA_LOCKED	Increments on a stream media clock locking.	Listener
MEDIA_UNLOCKED	Increments on a stream media clock unlocking.	Listener
STREAM_INTERRUPTED	Increments when stream playback is interrupted.	Talker/Listener
SEQ_NUM_MISMATCH	Increments when an Audio/Video Transport Protocol Data Unit (AVTPDU) is received with a nonsequential **sequence_num** field.	Listener
MEDIA_RESET	Increments on a toggle of the **mr** bit in the AVTPDU.	Talker/Listener
TIMESTAMP_UNCERTAIN	Increments on a toggle of the **tu** bit in the AVTPDU.	Talker/Listener
TIMESTAMP_VALID	Increments on receipt or transmit of an AVTPDU with the **tv** bit set.	Talker/Listener
TIMESTAMP_NOT_VALID	Increments on receipt or transmit of an AVTPDU with the **tv** bit cleared.	Talker/Listener
UNSUPPORTED_FORMAT	Increments on receipt of an AVTPDU that contains an unsupported media type.	Listener
LATE_TIMESTAMP	Increments on receipt of an AVTPDU with an **avtp_timestamp** field that is in the past.	Listener
EARLY_TIMESTAMP	Increments on receipt of an AVTPDU with an **avtp_timestamp** field that is too far in the future to process.	Listener
FRAMES_RX	Increments on each AVTPDU received.	Listener
FRAMES_TX	Increments on each AVTPDU transmited.	Talker

Copyright © 2016 IEEE. All rights reserved.

E.2 AVTP counter definitions

E.2.1 MEDIA_LOCKED

The MEDIA_LOCKED counter increments on a media clock lock event. A media clock is locked when the device is capable of using the media clock to render stream data. The specifics of a media clock lock event are implementation dependent and outside the scope of this standard.

E.2.2 MEDIA_UNLOCKED

The MEDIA_UNLOCKED counter increments on a media clock unlock event. A media clock is unlock when the device is not capable of using the media clock to render stream data. The specifics of a media clock unlock event are implementation dependent and outside the scope of this standard.

E.2.3 STREAM_INTERRUPTED

The STREAM_INTERRUPTED counter increments when the stream playback is interrupted. A stream is interrupted when the stream playback stops for any reason other than normal startup and tear-down.

E.2.4 SEQ_NUM_MISMATCH

The SEQ_NUM_MISMATCH counter increments when an AVTPDU is received with a non-sequential **sequence_num** field.

E.2.5 MEDIA_RESET

The MEDIA_RESET counter increments when an AVTPDU is received or transmitted where the **mr** bit toggled as compared to the previous AVTPDU.

E.2.6 TIMESTAMP_UNCERTAIN

The TIMESTAMP_UNCERTAIN counter increments when an AVTPDU is received or transmitted where the **tu** bit toggled as compared to the previous AVTPDU.

E.2.7 TIMESTAMP_VALID

The TIMESTAMP_VALID counter increments on receipt or transmit of an AVTPDU with the **tv** bit set to one (1).

E.2.8 TIMESTAMP_NOT_VALID

The TIMESTAMP_NOT_VALID counter increments on receipt or transmit of an AVTPDU with the **tv** bit set to zero (0).

NOTE—When AAF is used in sparse timestamping mode (**sp** bit is set, see 7.2.4), then 7 out of 8 AVTPDUs will have the **tv** bit set to zero (0), resulting in the TIMESTAMP_NOT_VALID counter incrementing more quickly than with

Copyright © 2016 IEEE. All rights reserved.

other formats. It is up to a higher-level application to determine if this is the case and whether or not the current accumulation is correct behavior.

E.2.9 UNSUPPORTED_FORMAT

The UNSUPPORTED_FORMAT counter increments on a receipt of an AVTPDU that contains an unexpected stream data format by the rendering device.

E.2.10 LATE_TIMESTAMP

The LATE_TIMESTAMP counter increments on receipt of an AVTPDU that cannot be utilized by the device due to the **avtp_timestamp** field containing a timestamp that is in the past.

E.2.11 EARLY_TIMESTAMP

The EARLY_TIMESTAMP counter increments on receipt of an AVTPDU with an **avtp_timestamp** field that is too far in the future to be utilized. A timestamp is considered too far in the future if, for example, the device does not contain enough buffer memory to store the stream data. It is possible for other device specific reasons to exist for a timestamp to be considered too far in the future.

E.2.12 FRAMES_RX

The FRAMES_RX counter increments on receipt of an AVTPDU.

E.2.13 FRAMES_TX

The FRAMES_TX counter increments on transmit of an AVTPDU.

Copyright © 2016 IEEE. All rights reserved.

Annex F

(normative)

Protocol Implementation Conformance Statement (PICS) proforma[21]

F.1 Introduction

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

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

a) By the protocol implementor, as a checklist to reduce the risk of failure to conform to the standard through oversight.

b) By the supplier and acquirer—or potential acquirer—of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma.

c) By the user—or potential user—of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that although interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSs).

d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

F.2 Abbreviations and special symbols

F.2.1 Status symbols

M Mandatory

O Optional

O.n Optional, but support of at least one of the group of options labeled by the same numeral n is required

X Prohibited predicate: Conditional-item symbol, including predicate identification (see F.3.4)

¬ Logical negation, applied to a conditional item's predicate

F.2.2 General abbreviations

N/A Not applicable

PICS Protocol Implementation Conformance Statement

[21] *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.

Copyright © 2016 IEEE. All rights reserved.

F.3 Instructions for completing the PICS proforma

F.3.1 General structure of the PICS proforma

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

The main part of the PICS proforma is a fixed-format questionnaire, divided into several subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the right-most column, either by simply marking an answer to indicate a restricted choice (usually Yes or No), or by entering a value or a set or range of values. (Note that there are some items in which two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered; the third column records the status of the item—whether support is mandatory, optional, or conditional; see also F.3.4. The fourth column contains the reference or references to the material that specifies the item in the main body of this standard, and the fifth column provides the space for the answers.

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

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

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

F.3.2 Additional information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing on the answers to some items.

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

F.3.3 Exception information

It may occasionally happen that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this; instead, the supplier shall write the missing answer into the

Copyright © 2016 IEEE. All rights reserved.

Support column, together with an Xi reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item.

An implementation for which an Exception item is required in this way does not conform to this standard.

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

F.3.4 Conditional status

F.3.4.1 Conditional items

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

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the "Not Applicable" (N/A) answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form "pred: S," where "pred" is a predicate as described in F.3.4.2, and "S" is a status symbol, either "M" or "O."

If the value of the predicate is true (see F.3.4.2), the conditional item is applicable, and its status is indicated by the status symbol following the predicate: the answer column is to be marked in the usual way. If the value of the predicate is false, the "Not Applicable" answer is to be marked.

F.3.4.2 Predicates

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma: The value of the predicate is true if the item is marked as supported, and is false otherwise.

b) A predicate-name, for a predicate defined as a Boolean expression constructed by combining item-references using the Boolean operator OR: The value of the predicate is true if one or more of the items is marked as supported.

c) A predicate-name, for a predicate defined as a Boolean expression constructed by combining item-references using the Boolean operator AND: The value of the predicate is true if all of the items are marked as supported.

d) The logical negation symbol "¬" prefixed to an item-reference or predicate-name: The value of the predicate is true if the value of the predicate formed by omitting the "¬" symbol is false, and vice versa.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

Copyright © 2016 IEEE. All rights reserved.

F.4 PICS proforma for IEEE Std 1722

F.4.1 Implementation identification

Table F.1—Identification

Supplier	___
Contact point for queries about the PICS	___
Implementation name(s) and version(s)	___
Other information necessary for full identification, e.g., name(s) and version(s) of machines and/or operating system names	___
NOTE 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification. NOTE 2—The terms *name* and *version* should be interpreted appropriately to correspond with a supplier's terminology (e.g., type, series, model).	

F.4.2 Protocol summary, IEEE Std 1722

Table F.2—Protocol summary

Identification of protocol specification	IEEE Std 1722, IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	___
Have any Exception items been required? (See F.3.3: the answer "Yes" means that the implementation does not conform to IEEE Std 1722)	No [] Yes []

Table F.3—Date of statement

Date of statement	___

Copyright © 2016 IEEE. All rights reserved.

F.5 Major capabilities and options

Table F.4—Major capabilities

Item	Feature	Status	References	Support
AVTP	Does this device support the Audio Video Transport Protocol (AVTP)?	M	Clause 4	Yes [ ]
61883	Does this device support IEC 61883–based transport?	O	Clause 5	Yes [ ] No [ ]
IIDC	Does this device support IIDC-based transport?	O	Clause 5	Yes [ ] No [ ]
MMA	Does this device support MMA Format?	O	Clause 6	Yes [ ] No [ ]
AAF	Does this device support AVTP Audio Format (AAF)?	O	Clause 7	Yes [ ] No [ ]
CVF	Does this device support Compressed Video Format (CVF)?	O	Clause 8	Yes [ ] No [ ]
ACF	Does this device support AVTP Control Format (ACF)?	O	Clause 9	Yes [ ] No [ ]
CRF	Does this device support Clock Reference Format (CRF)?	O	Clause 10	Yes [ ] No [ ]
SVF	Does this device support Serial Digital Interface (SDI) Video Format?	O	Clause 11	Yes [ ] No [ ]
RVF	Does this device support Raw Video Format (RVF)?	O	Clause 12	Yes [ ] No [ ]
AEF	Does this device support Advanced Encryption Standard (AES) Encrypted Format?	O	Clause 13	Yes [ ] No [ ]
VSF	Does this device support Vendor Specific Format (VSF)?	O	Clause 14	Yes [ ] No [ ]
ESCF	Does this device support Elliptic Curve Cryptography (ECC) Signed Control Format?	O	Clause 16	Yes [ ] No [ ]
EECF	Does this device support ECC Encrypted Control Format (EECF)?	O	Clause 17	Yes [ ] No [ ]
MAAP	Does this device support media access control (MAC) Address Acquisition Protocol (MAAP)?	O	Annex B	Yes [ ] No [ ]
IPENC	Does this device support encapsulating AVTP in User Datagram Protocol (UDP)?	O	Annex J	Yes [ ] No [ ]

F.6 AVTP Protocol

Table F.5—AVTP Protocol

Item	Feature	Status	References	Support
AVTP-1	Does the device transfer control of the AVTP frame to the MAC layer no later than the AVTP Presentation Time – Max Transit Time?	M	4.3.3	Yes [ ]
AVTP-2	Is the device's timing uncertainty less than Max Timing Uncertainty defined in Table 4?	M	4.3.3	Yes [ ]
AVTP-3	Do all AVTP end stations that transmit or receive time-sensitive streams on a plug-and-play network support the standards listed in 4.3.6?	M	4.3.6	Yes [ ]
AVTP-4	Does this device use an **ethertype** value listed in Table 5?	¬IPENC: M	4.4.2	Yes [ ] N/A[ ]
AVTP-5	Does this device use the EF_STREAM or EF_CONTROL **subtype**?	X	4.4.3.2	No [ ]
AVTP-6	Does this device set the **version** field to zero (0)?	M	4.4.3.4	Yes [ ]
AVTP-7	Does this device ignore Audio/Video Transport Protocol Data Units (AVTPDUs) received with a nonzero version number?	M	4.4.3.4	Yes [ ]
AVTP-8	If the **sv** field is set to one (1), does the **stream_id** field contain the 8-octet stream ID?	M	4.4.4.8	Yes [ ]
AVTP-9	If the specified **subtype** uses a common stream header, is the **sv** field set to one (1)?	M	4.4.4.8	Yes [ ]
AVTP-10	Is the **mr** bit stable for a minimum of 8 AVTPDUs for a given stream?	M	4.4.4.3	Yes [ ]
AVTP-11	If the **tv** bit is one (1), does the **avtp_timestamp** field contain a valid timestamp?	M	4.4.4.5, 5.5.2	Yes [ ]
AVTP-12	Does the **sequence_num** field increment by 1?	M	4.4.4.6	Yes [ ]
AVTP-13	Does the **sequence_num** field wrap from 11111111_2 to 00000000_2?	M	4.4.4.6	Yes [ ]
AVTP-14	Does the Listener tolerate a starting **sequence_num** of any value?	M	4.4.4.6	Yes [ ]
AVTP-15	Does the device support set/reset the value in the **tu** field as described?	M	4.4.4.7	Yes [ ]
AVTP-16	If the **tv** bit is one (1), is the **avtp_timestamp** in units of nanoseconds?	M	4.4.4.9	Yes [ ]
AVTP-17	If the **tv** bit is one (1), is the **avtp_timestamp** maximum value $2^{32} - 1$ ns?	M	4.4.4.9	Yes [ ]
AVTP-18	If the **tv** bit is zero (0), does the device ignore the received **avtp_timestamp**?	M	4.4.4.9	Yes [ ]
AVTP-19	Is the **stream_data_length** field set to allow all stream data to fit within a single Ethernet frame?	M	4.4.4.10	Yes [ ]
AVTP-20	Is the **control_data_length** field set to allow all stream data to fit within a single Ethernet frame?	M	4.4.5.4	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.7 61883 Protocol

Table F.6—AVTP-61883 Protocol

Item	Feature	Status	References	Support
61883-1	Does the device use the common stream header?	61883 OR IIDC:M	4.4.4	Yes[ ]
61883-2	Is the **subtype** field set to 61883_IIDC?	61883 OR IIDC:M	4.4.3.2	Yes[ ]
61883-3	Is the **tag** field set to 00_2?	IIDC:M	5.2.4	Yes[ ]
61883-4	Is the **tag** field set to 01_2?	61883:M	5.2.4	Yes[ ]
61883-5	Is the **channel** field set to 31 when the stream originates on the network?	61883 OR IIDC:M	5.2.5	Yes[ ]
61883-6	Is the **tcode** field set to 1010_2?	61883 OR IIDC:M	5.2.6	Yes[ ]
61883-7	Does the device ignore the **tcode** field on receive?	61883 OR IIDC:M	5.2.6	Yes[ ]
61883-8	Does every IIDC AVTPDU of an IIDC video frame have the same AVTP Presentation Time?	IIDC:M	5.3.3	Yes[ ]
61883-9	Is the 2-bit common isochronous packet (CIP) header first quadlet indicator set to 00_2?	61883:M	5.4.3.2	Yes[ ]
61883-10	Does the device ignore the 2-bit CIP header first quadlet indicator on receive?	61883:M	5.4.3.2	Yes[ ]
61883-11	Is the 6-bit **SID** field set to 63 when the stream originates on the AVTP network?	61883:M	5.4.3.3	Yes [ ]
61883-12	Is the **DBS** field set to a value that will not cause the AVTPDU to exceed the maximum transmission unit (MTU) of the network?	61883:M	5.4.3.4	Yes [ ]
61883-13	Does the AVTPDU contain an integral number of data blocks?	61883:M	5.4.3.4	Yes [ ]
61883-14	Is the 2-bit CIP header second quadlet indicator set to 10_2?	61883:M	5.4.3.9	Yes [ ]
61883-15	If **SPH** is zero (0), is the **SYT** field set to $FFFF_{16}$?	61883:M	5.4.4.4	Yes [ ]
61883-16	If **SPH** is zero (0), does the device ignore the **SYT** field on receive?	61883:M	5.4.4.4	Yes [ ]
61883-17	If the **SPH** is one (1), is the **tv** field set to zero?	61883:M	5.4.5.3, 61883-4, 61883-7	Yes [ ]
61883-18	If the data to be transmitted is larger than a standard Ethernet frame, are multiple CIP packets generated, each of which can fit in a standard Ethernet frame?	61883:M	—	Yes [ ]
61883-19	If using 61883-6 or 61883-8 encapsulation, does the transmitting device specify the AVTP Presentation Time of the event to the Listener?	61883:M	5.4.2.3, 61883-6, 5.4.2.5, 61883-8	Yes [ ]
61883-20	If using 61883-6 or 61883-8 encapsulation, is the Listener capable of presenting events at the time specified by the Talker?	61883:M	5.4.2.3, 61883-6, 5.4.2.5, 61883-8	Yes [ ]
61883-21	If using 61883-6 encapsulation, does the Talker add TRANSFER_DELAY to the AVTP Presentation Time?	61883:M	5.4.2.3, 61883-6	Yes [ ]
61883-22	If using 61883-8 encapsulation, does the device transmit a single stream of video per AVTP stream?	61883:M	5.4.2.5, 61883-8	Yes [ ]
61883-23	If using 61883-8 encapsulation, does the device transmission of nonempty AVTPDUs meet the conditions specified in 5.4.2.5?	61883:M	5.4.2.5, 61883-8	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.8 AVTP Audio Format

Table F.7—AVTP Audio Format

Item	Feature	Status	References	Support
AAF-1	Does the device use the common stream header?	AAF:M	4.4.4	Yes [ ]
AAF-2	Is the **sv** field set to one (1)?	AAF:M	4.4.4	Yes [ ]
AAF-3	Is the **version** field set to zero (0)?	AAF:M	4.4.3.4	Yes [ ]
AAF-4	Is the **subtype** field set to AAF?	AAF:M	4.4.3.2	Yes [ ]
AAF-5	Is the **mr** field toggled when the device's media clock source has changed?	AAF:M	4.4.4.3	Yes [ ]
AAF-6	When the **mr** field is toggled, does it remain in its new state for a minimum of eight (8) AAF AVTPDUs for a given continuous stream?	AAF:M	4.4.4.3	Yes [ ]
AAF-7	If the **tv** field is one (1), does the AVTP Presentation Time in the **avtp_timestamp** field correspond to the first audio sample contained in the AAF AVTPDU?	AAF:M	4.4.4.5, 7.5	Yes [ ]
AAF-8	Is the **sequence_num** field incremented by one (1) with wrapping, on each subsequent AAF AVTPDU of a stream?	AAF:M	4.4.4.6	Yes [ ]
AAF-9	Is the **sequence_num** field value wrapped from 11111111_2 to 00000000_2 (FF_{16} to 00_{16})?	AAF:M	4.4.4.6	Yes [ ]
AAF-10	Is the **tu** field set to zero (0) when gPTP time is stable?	AAF:M	4.4.4.7	Yes [ ]
AAF-11	Does the **stream_id** field contain a valid stream ID?	AAF:M	4.4.4.8	Yes [ ]
AAF-12	Does the **stream_data_length** field contain the length (in octets) of the AAF AVTPDU payload such that the AAF AVTPDU can fit within a single standard-frame payload?	AAF:M	4.4.4.10	Yes [ ]
AAF-13	Is the **format** field set to one of the values in Table 9?	AAF:M	7.2.3	Yes [ ]
AAF-14	Is the **nsr** field set to one of the values in Table 11?	AAF:M	7.3.2	Yes [ ]
AAF-15	Is the **channels_per_frame** field set to the number of audio channels represented in the audio sample frame, a positive integer greater than zero (0)?	AAF:M	7.3.3	Yes [ ]
AAF-16	If **format** is set to a non-integer type, is the **bit_depth** field set to zero (0)?	AAF:M	7.3.4	Yes [ ]
AAF-17	If **format** is set to an integer type, does the **bit_depth** field indicate how many bits within the sample contain valid data, a value less than or equal to the size of the format set in the **format** field?	AAF:M	7.3.4	Yes [ ]
AAF-18	If **format** is set to an integer type, and all bits are valid, is the **bit_depth** field set to zero (0)?	AAF:M	7.3.4	Yes [ ]
AAF-19	If **format** is set to an integer type, then in the **bit_depth** field value, are all unused bits set to zero (0)?	AAF:M	7.3.4	Yes [ ]
AAF-20	If **sp** field is zero (0), is the **tv** field set to one (1)?	AAF:M	7.2.4, 7.5	Yes [ ]
AAF-21	If **sp** field is one (1), is the **tv** field set to one (1) in every eighth AAF AVTPDU and zero (0) in all other AAF AVTPDUs in the stream?	AAF:M	7.2.4, 7.5	Yes [ ]
AAF-22	Do all AAF AVTPDUs in the stream contain the same number of audio sample frames per AAF AVTPDU?	AAF:M	7.6	Yes [ ]
AAF-23	If **channels_per_frame** is greater than one (1), then are the audio samples grouped chronologically into audio sample frames?	AAF:M	Figure 28, Figure 29	Yes [ ]
AAF-24	If **format** is set to AES3_32BIT, are the subframe quadlets sequenced as shown in Figure 33?	AAF:M	Figure 33	Yes [ ]
AAF-25	If **format** is set to AES3_32BIT, is the **nfr** field set to the AES3 frame rate?	AAF:M	7.4.2	Yes [ ]
AAF-26	If **format** is set to AES3_32BIT, is the **streams_per_frame** field set to the number of AES3 streams being transmitted, multiplied by 2?	AAF:M	7.4.3	Yes [ ]
AAF-27	If **format** is set to AES3_32BIT, are the AES3 subframes formatted according to Figure 34?	AAF:M	Figure 34, 7.4	Yes [ ]
AAF-28	If **format** is set to AES3_32BIT, are the AES3 subframes bits 28 to 31 (marked **reserved**) set to zero (0)?	AAF:M	Figure 34, 7.4	Yes [ ]
AAF-29	If **format** is set to AES3_32BIT, are the subframes sequenced as subframe 1 followed by subframe 2 for each AES3 frame?	AAF:M	Figure 33, 7.4	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

Table F.8—AVTP Audio Format *(continued)*

Item	Feature	Status	References	Support
AAF-30	If **format** is set to AES3_32BIT, does the bit order of the audio sample word within the AES3 subframe follow the AVTP convention (most-significant bit at least offset)?	AAF:M	Figure 1, 7.4	Yes [ ]
AAF-31	If **format** is set to AES3_32BIT, are the audio sample word and ancillary data bits aligned as shown in Figure 34?	AAF:M	Figure 34, 7.4	Yes [ ]
AAF-32	If **format** is set to AES3_32BIT, are all bits of the AES3 24-bit audio sample word field transmitted without modification?	AAF:M	7.4, K.10.7	Yes [ ]
AAF-33	If **format** is set to AES3_32BIT, is the **B** bit set to one (1) for the first subframe of the block specified in AES3-3, and to zero (0) for all other subframes?	AAF:M	7.4.6.3	Yes [ ]
AAF-34	If **format** is set to AES3_32BIT, does the **C** bit contain the channel status content bit (AES3-2-2009, Section 5.5 Channel status content)?	AAF:M	7.4.6.3	Yes [ ]
AAF-35	If the **format** is set to AES3_32BIT and the AES3 samples are PCM, is the **V** bit set to zero (0)?	AAF:M	7.4.6.3, K.5	Yes [ ]
AAF-36	If the **format** is set to AES3_32BIT and the AES3 samples are not PCM, is the **V** bit set to one (1)?	AAF:M	7.4.6.3, K.5	Yes [ ]
AAF-37	If **format** is set to AES3_32BIT, is the **aes3_dt_ref** field set to one of the values in Table 16?	AAF:M	7.4.5	Yes [ ]
AAF-38	If **format** is set to AES3_32BIT and **aes3_dt_ref** is set to DT_UNSPECIFIED, is the **aes3_data_type** field set to zero (0)?	AAF:M	7.4.4	Yes [ ]
AAF-39	If **format** is set to AES3_32BIT and **aes3_dt_ref** is set to DT_PCM, is the **aes3_data_type** field set to zero (0)?	AAF:M	7.4.4	Yes [ ]
AAF-40	If **format** is set to AES3_32BIT and **aes3_dt_ref** is set to DT_SMPTE338, is the **aes3_data_type** field set to a valid entry from the SMPTE ST 338:2015, Section 4, Data Types, Table 1?	AAF:M	7.4.4	Yes [ ]
AAF-41	If **format** is set to AES3_32BIT and **aes3_dt_ref** is set to DT_IEC61937, is the **aes3_data_type** field set to a valid entry from IEC 61937-2:2011 Amend. 1, Section 4.2, Table 2?	AAF:M	7.4.4	Yes [ ]
AAF-42	If the AVTP Audio Stream is either stereo or 5.1 or 7.1 format, are the stream channels mapped according to Table 12, Table 13, or Table 14?	AAF:M	7.3.6	Yes [ ]

F.9 MMA Format

Table F.9—MMA Format

Item	Feature	Status	References	Support
MMA-1	Does the device use the common stream header?	MMA:M	4.4.4	Yes [ ]
MMA-2	Is the **subtype** field set to MMA_STREAM?	MMA:M	4.4.3.2	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.10 Compressed Video Format

Table F.10—Compressed Video Format

Item	Feature	Status	References	Support
CVF-1	Does the device use the common stream header?	CVF:M	4.4.4	Yes [ ]
CVF-2	Is the **subtype** field set to CVF?	CVF:M	4.4.3.2	Yes [ ]
CVF-3	Is the **format** field set to a non-reserved value from Table 19?	CVF:M	8.3.2	Yes [ ]
CVF-4	Is the **format_subtype** field set to a non-reserved value from Table 20?	CVF:M	8.3.3	Yes [ ]

Table F.11—MJPEG Compressed Video Format

Item	Feature	Status	References	Support
MJPEG-1	Is the **avtp_timestamp** set and valid (tv = 1) for the AVTPDU carrying a final fragment of a video packetization unit? Is the **avtp_timestamp** for all AVPTDUs carrying pre-final fragments set to invalid (tv = 0)?	CVF and MJPEG:M	8.4.2.2, 8.4.4	Yes [ ]
MJPEG-2	Is the **M** bit set in the last packet of a video frame?	CVF and MJPEG:M	8.4.2.3, 8.4.4	Yes [ ]
MJPEG-3	Is the **fragment_offset** field set to the byte offset of its payload data within the original video frame?	MJPEG:M	8.4.4	Yes [ ]
MJPEG-4	Is the video frame data contained in a given CVF AVTPDU from a single JPEG video frame?	CVF and MJPEG:M	8.4.4	Yes [ ]

Table F.12—H.264 Compressed Video Format

Item	Feature	Status	References	Support
H264-1	Is the **avtp_timestamp** field set as described in 8.5.2.2?	CVF and H264:M	8.5.2.2	Yes [ ]
H264-2	Is the **ptv** bit set as described in 8.5.2.3?	H264:M	8.5.2.3	Yes [ ]
H264-3	Is the **M** bit set in the last packet of the access unit indicated by the **avtp_timestamp**?	CVF and H264:M	8.5.2.4	Yes [ ]
H264-4	Is the **M** bit set in the CVF AVTPDU that contains the last Network Abstraction Layer (NAL) unit in an aggregation packet?	CVF and H264:M	8.5.2.4	Yes [ ]
H264-5	Is the **h264_timestamp** field populated according to 8.5.3.1?	H264:M	8.5.3.1	Yes [ ]
H264-6	Is the **h264_payload** field populated according to 8.5.4?	H264:M	8.5.4	Yes [ ]

Table F.13—JPEG-2000 Compressed Video Format

Item	Feature	Status	References	Support
J2000-1	Is the **M** bit set to one (1) in the last CVF AVTPDU of a video frame?	CVF and J2000:M	8.6.2.1	Yes [ ]
J2000-2	Is the **M** bit set to zero (0) in all CVF AVTPDUs of a video frame other than the last CVF AVTPDU?	CVF and J2000:M	8.6.2.1	Yes [ ]
J2000-3	Is the **avtp_timestamp** field set according to 8.6.2.2?	CVF and J2000:M	8.6.2.2	Yes [ ]
J2000-4	If multiple packetization units are in a single CVF AVTPDU, is codestream order preserved?	CVF and J2000:M	8.6.4.2	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.11 AVTP Control Format

Table F.14—AVTP Control Format

Item	Feature	Status	References	Support
	Does the device use the alternative header for Non–Time-Synchronous Control Format (NTSCF) ACF AVTPDUs?	NTSCF:M	9.2	Yes [ ]
	Does the device set the **subtype** field to NTSCF?	NTSCF:M	9.2	Yes [ ]
	Does the device produce NTSCF AVTPDUs with the NTSCF header fields formatted as described in 9.2?	NTSCF:M	9.2	Yes [ ]
	Does the **sequence_num** field increment by one (1) with wrapping?	NTSCF:M	9.2.4	Yes [ ]
	Is a Listener able to tolerate a starting **sequence_num** of any value?	NTSCF:M	9.2.4	Yes [ ]
	Does the **stream_id** contain a valid StreamID, as defined in subclause 35.2.2.8.2 of IEEE Std 802.1Q-2014)?	NTSCF:M	9.2.5	Yes [ ]
	If the **sv** field has a value of zero (0), does the **stream_id** contain an Extended Unique Identifier (EUI-64)?	NTSCF:M	9.2.5	Yes [ ]
	Does the device use the common stream header for TSCF ACF AVTPDUs?	TSCF:M	9.3	Yes [ ]
	For TSCF ACF AVTPDUs: is the **subtype** field set to TSCF?	TSCF:M	9.3	Yes [ ]
	Is the **acf_msg_type** field set to ACF_FLEXRAY?	ACF_FLEXRAY:M	9.4.2	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_FLEXRAY:M	9.4.2	Yes [ ]
	Are all the fields in the FLEXRAY ACF message set as described in the referenced clause?	ACF_FLEXRAY:M	9.4.2	Yes [ ]
	Is the **acf_msg_type** field set to ACF_CAN?	ACF_CAN:M	9.4.3	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_CAN:M	9.4.3	Yes [ ]
	Are all the fields in the CAN ACF message set as described in the referenced clause?	ACF_CAN:M	9.4.3	Yes [ ]
	Is the **acf_msg_type** field set to ACF_CAN_BRIEF?	ACF_CAN_BRIEF:M	9.4.4	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_CAN_BRIEF:M	9.4.4	Yes [ ]
	Are all the fields in the CAN_BRIEF ACF message set as described in the referenced clause?	ACF_CAN_BRIEF:M	9.4.4	Yes [ ]
	Is the **acf_msg_type** field set to ACF_LIN?	ACF_LIN:M	9.4.5	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_LIN:M	9.4.5	Yes [ ]
	Are all the fields in the LIN ACF message set as described in the referenced clause?	ACF_LIN:M	9.4.5	Yes [ ]
	Is the **acf_msg_type** field set to ACF_MOST?	ACF_MOST:M	9.4.6	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_MOST:M	9.4.6	Yes [ ]
	Are all the fields in the MOST ACF message set as described in the referenced clause?	ACF_MOST:M	9.4.6	Yes [ ]
	Is the **acf_msg_type** field set to ACF_GPC?	ACF_GPC:M	9.4.7	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_GPC:M	9.4.7	Yes [ ]
	Are all the fields in the GPC ACF message set as described in the referenced subclause?	ACF_GPC:M	9.4.7	Yes [ ]
	Is the **acf_msg_type** field set to ACF_SERIAL?	ACF_SERIAL:M	9.4.8	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_SERIAL:M	9.4.8	Yes [ ]
	Are all the fields in the SERIAL ACF message set as described in the referenced clause?	ACF_SERIAL:M	9.4.8	Yes [ ]
	Is the **sensor_msg_payload** field zero-padded as described in 9.4.10?	ACF_SERIAL:M	9.4.8	Yes [ ]
	Is the **acf_msg_type** field set to ACF_PARALLEL?	ACF_PARALLEL:M	9.4.9	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_PARALLEL:M	9.4.9	Yes [ ]
	Are all the fields in the PARALLEL ACF message set as described in the referenced clause?	ACF_PARALLEL:M	9.4.9	Yes [ ]
	Is the **acf_msg_type** field set to ACF_SENSOR?	ACF_SENSOR:M	9.4.10	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_SENSOR:M	9.4.10	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

Table F.15—AVTP Control Format *(continued)*

Item	Feature	Status	References	Support
	Are all the fields in the SENSOR ACF message set as described in the referenced clause?	ACF_SENSOR:M	9.4.10	Yes [ ]
	Is the **acf_msg_type** field set to ACF_SENSOR_BRIEF?	ACF_SENSOR_BRIEF:M	9.4.11	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_SENSOR_BRIEF:M	9.4.11	Yes [ ]
	Are all the fields in the SENSOR_BRIEF ACF message set as described in the referenced clause?	ACF_SENSOR_BRIEF:M	9.4.11	Yes [ ]
	Is the **acf_msg_type** field set to ACF_AECP?	ACF_AECP:M	9.4.12	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_AECP:M	9.4.12	Yes [ ]
	Are all the fields in the AECP ACF message set as described in the referenced clause?	ACF_AECP:M	9.4.12	Yes [ ]
	Is the **acf_msg_type** field set to ACF_ANCILLARY?	ACF_ANCILLARY:M	9.4.13	Yes [ ]
	Is the **acf_msg_length** field set according to 9.4.1.3?	ACF_ANCILLARY:M	9.4.13	Yes [ ]
	Are all the fields in the ANCILLARY ACF message set as described in the referenced clause?	ACF_ANCILLARY:M	9.4.13	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.12 Clock Reference Format

Table F.16—Clock Reference Format

Item	Feature	Status	References	Support
CRF-1	Does the device use the alternative header?	CRF:M	4.4.6	Yes [ ]
CRF-2	Is the **subtype** field set to CRF?	CRF:M	4.4.3.2	Yes [ ]
CRF-3	Is the **mr** bit set as described in 10.4.3?	CRF:M	10.4.3	Yes [ ]
CRF-4	Does a CRF Listener that is also a media stream Talker toggle the **mr** bit in outgoing streams as described in 10.4.3?	CRF:M	10.4.3	Yes [ ]
CRF-5	Does the **mr** bit remain in its new state for a minimum of 8 CRF AVTPDUs?	CRF:M	10.4.3	Yes [ ]
CRF-6	Is the **sequence_num** field increment by one (1) with wrapping?	CRF:M	10.4.6	Yes [ ]
CRF-7	Is a Listener able to tolerate a starting **sequence_num** of any value?	CRF:M	10.4.6	Yes [ ]
CRF-8	Is the **base_frequency_field** set to a value from 1 to 536 870 911 ($1FFFFFFF_{16}$)?	CRF:M	10.4.10	Yes [ ]
CRF-9	Is the **crf_data_length** field value a multiple of 8 octets?	CRF:M	10.4.11	Yes [ ]
CRF-10	Is the **timestamp_interval** nonzero?	CRF:M	10.4.12	Yes [ ]
CRF-11	Is the **fs** bit set to zero (0) for CRF AVTPDUs with a **type** field set to CRF_AUDIO_SAMPLE?	CRF:M	10.4.13.3	Yes [ ]
CRF-12	Is the **fs** bit set to zero (0) for CRF AVTPDUs with a **type** field set to CRF_VIDEO_SAMPLE?	CRF:M	10.4.13.4	Yes [ ]
CRF-13	Is the **fs** set to one (1) for CRF AVTPDUs with a **type** field set to CRF_VIDEO_LINE and the timestamp represents the first line of a video frame?	CRF:M	10.4.13.5	Yes [ ]
CRF-14	Is the **fs** set to zero (0) for CRF AVTPDUs with a **type** field set to CRF_VIDEO_LINE and the timestamp does not represent the first line of a video frame?	CRF:M	10.4.13.5	Yes [ ]
CRF-15	Is a CRF Audio Listener capable of receiving CRF AVTPDUs at the rates given in Table 28?	CRF:O	10.5	Yes [ ] No [ ]
CRF-16	Is a CRF Video Frame Listener capable of receiving CRF AVTPDUs at the rates given in Table 29?	CRF:O	10.5	Yes [ ] No [ ]
CRF-17	Is a CRF Video Line Listener capable of receiving CRF AVTPDUs at the rates given in Table 30?	CRF:O	10.5	Yes [ ] No [ ]
CRF-18	Is a CRF Video Line Listener capable of receiving CRF AVTPDUs at the rates given in Table 31?	CRF:O	10.5	Yes [ ] No [ ]
CRF-19	Is the CRF timestamp contained in the **crf_data** field set to a value which is offset, in positive nanoseconds, from the transmit reference plane at the Talker by the Max Transit Time of the network, rounded up to the nearest multiple of the media clock as shown in Equation (14)?	CRF:M	10.7	Yes [ ]
CRF-20	In a CRF AVTPDU that contains multiple CRF timestamps, do the values of the timestamps increase monotonically, and with a constant nominal spacing in time, from the first timestamp to the last timestamp in the CRF AVTPDU when **tu** equals zero (0)?	CRF:M	10.7	Yes [ ]
CRF-21	In a CRF AVTPDU that contains multiple CRF timestamps, are all timestamps derived from a continuous gPTP clock reference?	CRF:M	10.7	Yes [ ]
CRF-22	Are all **avtp_timestamps** sent by CRF Talkers within +/− 5.0% of the media sample period of the received CRF stream?	CRF:M	10.8 Equation (15)	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.13 SDI Video Format

Table F.17—SDI Video Format

Item	Feature	Status	References	Support
SVF-1	Does the device use the common stream header?	SVF:M	4.4.4	Yes[ ]
SVF-2	Is the **subtype** field set to SVF?	SVF:M	4.4.3.2	Yes[ ]
SVF-3	Is the **tv** field set to one (1) in the SVF AVTPDU containing the first packet of each video line?	SVF:M	11.2.2	Yes [ ]
SVF-4	Does the first packet of each video line contain a valid timestamp in the **avtp_timestamp** field?	SVF:M	11.2.2	Yes [ ]
SVF-5	Is the **tv** field set to zero (0) for all SVF AVTPDUs that do not contain the first packet of a video line?	SVF:M	11.2.2	Yes [ ]
SVF-6	If the **format** field is set to SVF_USER, do the **map**, **sample**, **frame**, and **frate** fields contain correct values for the stream?	SVF:M	Figure 77, Table 32	Yes [ ]
SVF-7	If the **format** field is not set to SVF_USER, do the **map**, **sample**, **frame**, and **frate** fields contain values from Table 33?	SVF:M	11.2.2	Yes [ ]
SVF-8	If the **map**, **sample**, **frame**, and **frate** fields contain values from Table 33, are they ignored on receive?	SVF:M	11.2.2	Yes [ ]
SVF-9	Is the **i_seq_num** field set according to 11.2.4?	SVF:M	11.2.4	Yes [ ]
SVF-10	Is the **stream_data_length** set corresponding to the **format** field as specified in Table 34?	SVF:M	11.2.6, Table 34	Yes [ ]
SVF-11	Is the **sp** field set to one (1) for the last SVF AVTPDU of a meta-frame?	SVF:M	11.2.8	Yes [ ]
SVF-12	Is the **sp** field set to zero (0) for all SVF AVTPDUs other than the last SVF AVTPDU of a meta-frame?	SVF:M	11.2.8	Yes [ ]
SVF-13	Is the **ef** field set to one (1) for the last SVF AVTPDU of a video frame?	SVF:M	11.2.9	Yes [ ]
SVF-14	Is the **ef** field set to zero (0) for all SVF AVTPDUs other than the last SVF AVTPDU of a video frame?	SVF:M	11.2.9	Yes [ ]
SVF-15	Is the **map** field set to one of the values defined in Table 35?	SVF:M	11.2.11, Table 35	Yes [ ]
SVF-16	Is the **sample** field set to one of the values defined in Table 36?	SVF:M	11.2.12, Table 36	Yes [ ]
SVF-17	Is the **frame** field set to one of the values defined in Table 37?	SVF:M	11.2.13, Table 37	Yes [ ]
SVF-18	Is the **frate** field set to one of the values defined in Table 38?	SVF:M	11.2.14, Table 38	Yes [ ]
SVF-19	Does the **frcount** field increment by one (1) at the start of each video frame?	SVF:M	11.2.15	Yes [ ]
SVF-20	Is the first data in the **video_data_payload** in the first SVF AVTPDU of a video line the code for End of Active Video?	SVF:M	11.2.16, Figure 77	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.14 Raw Video Format

Table F.18—Raw Video Format

Item	Feature	Status	References	Support
RVF-1	Does the device use the common stream header?	RVF:M	4.4.4	Yes[ ]
RVF-2	Is the **subtype** field set to RVF?	RVF:M	4.4.3.2	Yes[ ]
RVF-3	Is the **tv** field set to one (1) for every RVF AVTPDU in which **i_seq_num** and **line_number** are set to zero (0)?	RVF:M	12.2.2	Yes [ ]
RVF-4	Do all RVF AVTPDUs in which **i_seq_num** and **line_number** are set to zero (0) contain a valid timestamp in the **avtp_timestamp** field?	RVF:M	12.2.2	Yes [ ]
RVF-5	Is the **tv** field set to zero (0) for all RVF AVTPDUs where **i_seq_num** and **line_number** are not set to zero (0)?	RVF:M	12.2.2	Yes [ ]
RVF-6	Is the number of active pixels the same for every line through the life of the stream?	RVF:M	12.2.3	Yes [ ]
RVF-7	Is the **stream_data_length** field constant over the duration of the stream?	RVF:M	12.2.19, 12.2.20	Yes [ ]
RVF-8	Is the **stream_data_length** field set according to the Equation (17) and Equation (18)?	RVF:M	Equation (17) Equation (18)	Yes [ ]
RVF-9	Is the **f** field set to zero (0) for all RVF AVTPDUs of Field 1?	RVF:M	12.2.6	Yes [ ]
RVF-10	Is the **f** field set to one (1) for all RVF AVTPDUs of Field 2?	RVF:M	12.2.6	Yes [ ]
RVF-11	Is the **f** field set to zero (0) for all RVF AVTPDUs of a non-interlaced stream?	RVF:M	12.2.6	Yes [ ]
RVF-12	Is the **ef** field set to one (1) for the last RVF AVTPDU of a video frame?	RVF:M	12.2.7	Yes [ ]
RVF-13	Is the **ef** field set to zero (0) for all RVF AVTPDUs other than the last RVF AVTPDU of a video frame?	RVF:M	12.2.7	Yes [ ]
RVF-14	Is the **pixel_depth** field set to one of the values defined in Table 40?	RVF:M	12.2.11, Table 40	Yes [ ]
RVF-15	Is the **pixel_format** field set to one of the values defined in Table 41?	RVF:M	12.2.12, Table 41	Yes [ ]
RVF-16	Is the **frame_rate** field set to one of the values defined in Table 42?	RVF:M	12.2.13, Table 42	Yes [ ]
RVF-17	Is the **colorspace** field set to one of the values defined in Table 43?	RVF:M	12.2.14, Table 43	Yes [ ]
RVF-18	Is the **num_lines** field set to the number of full lines contained in the RVF AVTPDU?	RVF:M	12.2.14	Yes [ ]
RVF-19	Is the **num_lines** field set to zero (0) if the RVF AVTPDU contains only a portion of a line?	RVF:M	12.2.14	Yes [ ]
RVF-20	Is the **i_seq_num** field set according to raw_i_seq_num_field?	RVF:M	12.2.16	Yes [ ]
RVF-21	Is the **line_number** field set to the line number of the first line in the RVF AVTPDU?	RVF:M	12.2.17	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.15 AES Encrypted Format

Table F.19—AES Encrypted Format

Item	Feature	Status	References	Support
AEF-1	Is the **subtype** field set to AEF_CONTINUOUS when the encapsulated AEF AVTPDU has a subtype that uses a stream header?	AEF:M	13.2, 4.4.4	Yes [ ]
AEF-2	Is the **subtype** field set to AEF_DISCRETE when the encapsulated AEF AVTPDU has a subtype that uses a control or alternative header?	AEF:M	13.2, 4.4.5, 4.4.6	Yes [ ]
AEF-3	Is the **blocks** field set to minimum value of one (1) and a maximum value of $5B_{16}$?	AEF:M	13.2.3	Yes [ ]
AEF-4	Does the **encrypted_payload** field have a length that is an integer multiple of 16 octets?	AEF:M	13.2.5	Yes [ ]
AEF-5	Are AEF AVTPDUs that are indicated as not authentic by the AES-SIV algorithm discarded?	AEF:M	13.3.3	Yes [ ]

F.16 Vendor Specific Format

Table F.20—Vendor Specific Format

Item	Feature	Status	References	Support
VSF-1	Does the device use the common stream header for Vendor Stream Format?	VSF:M	4.4.4	Yes[ ]
VSF-2	Is the **subtype** field set to VSF_STREAM for Vendor Stream Format?	VSF:M	4.4.3.2	Yes[ ]
VSF-3	Do the **vendor_id_1** and **vendor_id_2** fields contain an MA-L (OUI), MA-M, or MA-S in the AVTP Vendor Specific Format?	VSF:M	14.1.1.2	Yes [ ]

F.17 ECC Signed Control Format

Table F.21—ECC Signed Control Format

Item	Feature	Status	References	Support
ESCF-1	Does the device use the alternative header?	ESCF:M	4.4.6	Yes [ ]
ESCF-2	Is the **subtype** field set to ESCF?	ESCF:M	4.4.3.2	Yes [ ]
ESCF-3	Are ESCF AVTPDUs with a **signed_timestamp** field value more that 100 milliseconds in the past discarded?	ESCF:M	16.3.3	Yes [ ]
ESCF-4	Are ESCF AVTPDUs with a **signed_timestamp** field value in the future discarded?	ESCF:M	16.3.3	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.18 ECC Encrypted Control Format

Table F.22—ECC Encrypted Control Format

Item	Feature	Status	References	Support
EECF-1	Does the device use the alternative header?	EECF:M	4.4.6	Yes [ ]
EECF-2	Is the **subtype** field set to EECF?	EECF:M	4.4.3.2	Yes [ ]
EECF-3	Is the secret value derivation primitive ECSFDP-DHC without compatibility with ECSVDP-DH?	EECF:M	17.3	Yes [ ]
EECF-4	Is the method for encrypting the message KDF2 using the hash function SHA-256?	EECF:M	17.3	Yes [ ]
EECF-5	Is the symmetric encryption scheme AES-CBC-IV0?	EECF:M	17.3	Yes [ ]
EECF-6	Is ECIES used in DHAES mode?	EECF:M	17.3	Yes [ ]
EECF-7	Are the elliptic curve primitives X-coordinate only representation EC2OSP-X and OS2ECP-X?	EECF:M	17.3	Yes [ ]
EECF-8	Are EECF AVTPDUs with an **encryption_timestamp** field value more that 100 milliseconds in the past discarded?	EECF:M	17.3	Yes [ ]
EECF-9	Are EECF AVTPDUs with an **encryption_timestamp** field value in the future discarded?	EECF:M	17.3	Yes [ ]

Copyright © 2016 IEEE. All rights reserved.

F.19 MAC Address Acquisition Protocol

Table F.23—AVTP MAAP

Item	Feature	Status	References	Support
MAAP-1	Does the device use addresses from the MAAP dynamic allocation pool?	MAAP: O	B.1, Table B.9	Yes [ ] No [ ]
MAAP-2	Is the SA MAC address set to the MAC address of the sender?	MAAP: M	B.2	Yes [ ]
MAAP-3	Is the **ethertype** set to the EtherType defined in Table 5?	MAAP: M	B.2	Yes [ ]
MAAP-4	Is the **control_data_length** set to 16_{10}?	MAAP: M	B.2	Yes [ ]
MAAP-5	Is the **subtype** field set to $7E_{16}$?	MAAP: M	B.2.2	Yes [ ]
MAAP-6	Is the **sv** field set to zero (0)?	MAAP: M	B.2.2	Yes [ ]
MAAP-7	Does the device ignore reserved message_types?	MAAP: M	B.2.2	Yes [ ]
MAAP-8	Does the device use the **version** field in accordance with B.2.3, B.2.3.2, and B.2.3.4?	MAAP: M	B.2.3	Yes [ ]
MAAP-9	Is the **stream_id** set to zero (0)?	MAAP: M	B.2.4	Yes [ ]
MAAP-10	Does the device implement the MAAP state machine as described in B.3.2?	MAAP: M	B.3.2	Yes [ ]
MAAP-11	Is the random number generator seeded in accordance with B.3.6.1?	MAAP: M	B.3.6.1	Yes [ ]
MAAP-12	Are MAC addresses compared in octet-wise reverse order?	MAAP: M	B.3.6.4	Yes [ ]
MAAP-13	Does the compare_MAC function return TRUE if the MAC address of the receiving station is lower than the MAC address from the received MAAP AVTPDU?	MAAP: M	B.3.6.4	Yes [ ]

F.20 IP Encapsulation

Table F.24—IP Encapsulation

Item	Feature	Status	References	Support
IPENC-1	Does **encapsulation_sequence_num** increment by one (1) with each IP AVTPDU with the same source IP address, destination IP address, source port, and destination port?	IPENC:M	J.3.2	Yes [ ]
IPENC-2	Does the **encapsulation_sequence_num** wrap from $FFFFFFFF_{16}$ to 00000000_{16}?	IPENC:M	J.3.2	Yes [ ]
IPENC-3	Does Listener of the IP AVTPDU stream tolerate a starting **encapsulation_sequence_num** of any value?	IPENC:M	J.3.2	Yes [ ]
IPENC-4	Does **encapsulated_avtpdu** field contain a valid AVTPDU as defined by Clause 4?	IPENC:M	J.3.3	Yes [ ]
IPENC-5	Is the source port of the IP AVTPDU set to an ephemeral port as defined by RFC6056?	IPENC:M	J.4	Yes [ ]
IPENC-6	If the encapsulation is Continuous, is the IP AVTPDU destination port set to 17220?	IPENC:O	J.5	Yes [ ] No [ ]
IPENC-7	If the encapsulation is Continuous, is the IP AVTPDU destination port set to 17221?	IPENC:M	J.5	No [ ]
IPENC-8	If the encapsulation is Discrete, is the IP AVTPDU destination port set to 17221?	IPENC:O	J.5	Yes [ ] No [ ]
IPENC-9	If the encapsulation is Discrete, is the IP AVTPDU destination port set to 17220?	IPENC:M	J.5	No [ ]

Copyright © 2016 IEEE. All rights reserved.

Annex G

(informative)

Example event field usage

G.1 Overview

The **evt** (event) field, which is found in Audio/Video Transport Protocol (AVTP) Audio Format (AAF) (Clause 7), Compressed Video Format (CVF) (Clause 7), SDI Video Format (SVF) (Clause 8), and Raw Video Format (RVF) (Clause 12), is reserved for application specific usage. This annex describes several possible usages of the **evt** field.

G.2 Synchronized format changes in a automotive environment

A high-end automotive audio system is an example of a system that can use the **evt** field to enable dynamic format changes. In this example, the vehicle's head unit is the audio source device that sends audio to a sink device or amplifier. The audio is formatted as either 1 channel (mono), 2 channels (stereo), 6 channels (surround sound), or 8 channels (surround sound) of 16-bit 44.1 or 48 kHz LPCM audio samples. The amplifier then uses a digital signal processor (DSP) to convert this audio stream into an appropriate format to drive a speaker array that can consist of as few as two, and as many as dozens, of speakers. The head unit often needs to change the format of the audio data on the fly when changes occur such as a switch from stereo radio to a surround-sound sound track for a movie. Creating one AVTP audio stream that is capable of transporting any of these formats without tearing down and creating a new stream is desirable to save switching time and improve the customer experience.

An example of a high-end automotive system is given in Figure G.1.

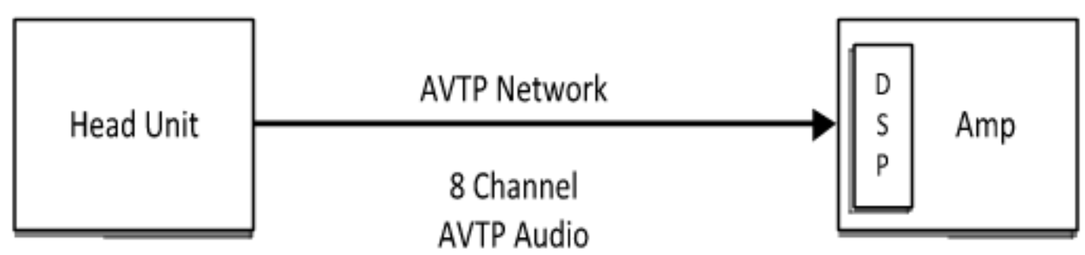


Figure G.1—Example high-end automotive system

This is accomplished by using a single 8-channel AVTP audio stream and indicating in real time which channels are active and the current format. The **evt** field is used to indicate the valid channels and the stream format as defined in 7.3.6. The evt field is set as described in Table G.1. The actual format of the AVTP stream remains consistent as an 8-channel audio stream, but the Talker only needs to populate the valid channels with data, while simply filling unused channels with zeroes.

Copyright © 2016 IEEE. All rights reserved.

Table G.1—Channel usage

Event bits	Audio format	Valid channels
0	Static	User defined
1	Mono	0
2	Stereo	0, 1
3	5.1	0, 1, 2, 3, 4, 5
4	7.1	0, 1, 2, 3, 4, 5, 6, 7
5 to 15	User defined	User defined

This use of the **evt** field allows for the amplifier to have knowledge of the current configuration on a sample-by-sample basis enabling it to react quickly and update the algorithms to provide a seamless switchover between channel arrangements. A seamless switchover would not be possible if a stream tear-down and creation were required.

G.3 Synchronized external event indication

Another possible usage for the **evt** field is in a professional audio environment to coordinate an audio stream with an external event. This allows coordination with other audio or video streams, or infrastructure systems. A secondary audio or video signal can begin to play on receipt of an event signal. In addition, the **evt** field could signal a lighting change, curtain open or close, or other infrastructure change.

Copyright © 2016 IEEE. All rights reserved.

Annex H

(informative)

Content protection

This annex is intended to provide a high-level overview of Time-Sensitive Networking (TSN) and content protection using High-bandwidth Digital Content Protection (HDCP) in IEEE Std 1722. Implementing HDCP over an Audio/Video Transport Protocol (AVTP) network requires the use of the HDCP Interface Independent Adaptation (IIA) as defined in Digital Content Protection [B6]. No details related to interpretation of HDCP IIA will be given. It is up to the implementer to determine that their specific implementation is in compliance with the HDCP IIA.

A high-level block diagram of an HDCP system is given in Figure H.1.

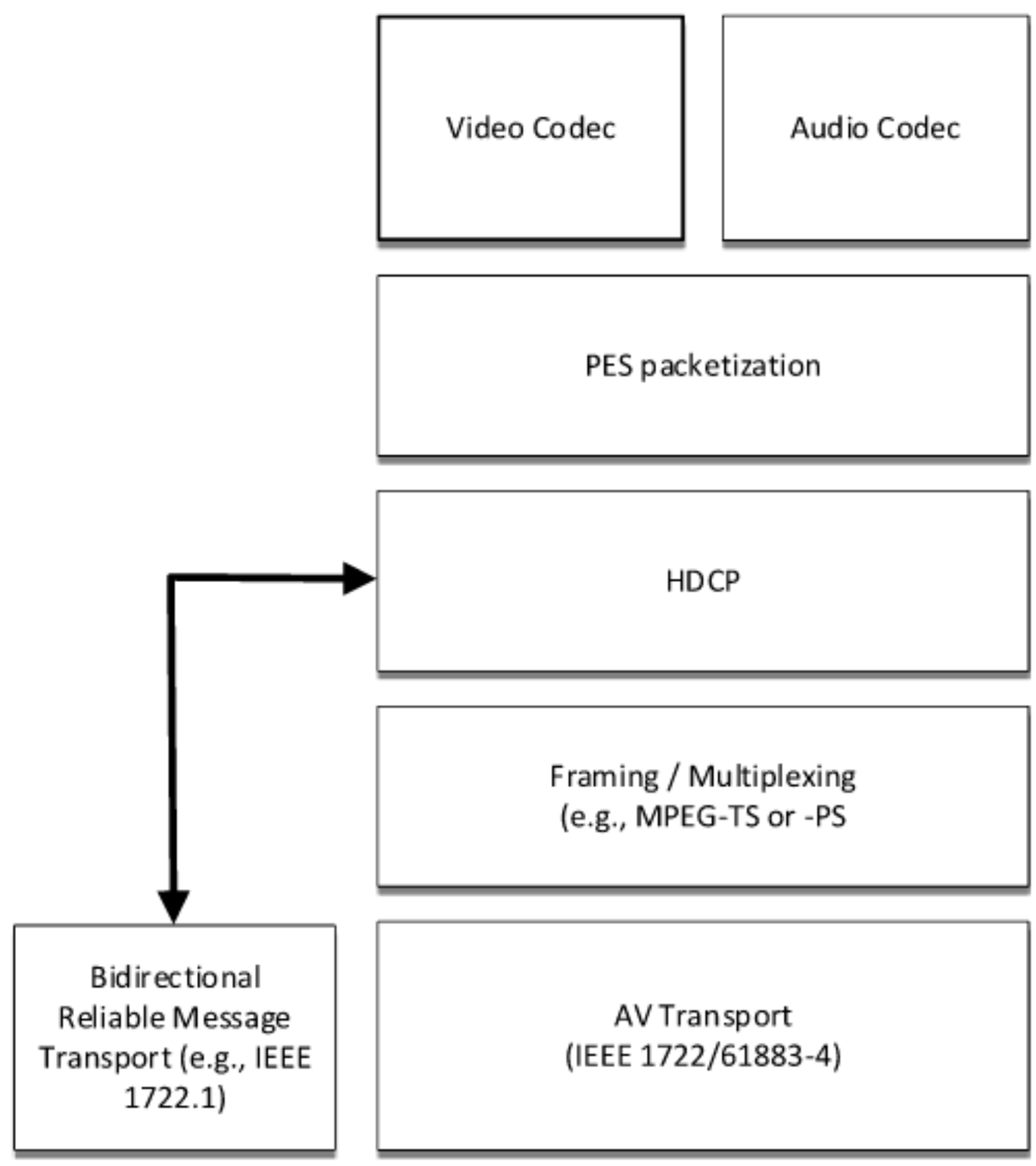


Figure H.1—HDCP block diagram

Support for HDCP over AVTP can be divided into two high-level functions. The first is transport of the actual media, the second is the exchange of HDCP IIA messages that do not contain media.

Transport of the HDCP-encrypted media is done using an MPEG Transport Stream as described in Section 3 of the HDCP IIA specification. Media data contained in an MPEG transport stream that has been encrypted is transported using AVTP using an IEC 61883-4 stream as defined in Clause 6.

A reliable packet protocol is required to transport HDCP Authentication Protocol Messages described in Section 4 of the HDCP IIA specification. In an AVTP network this transport is Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC) (See subclause 9.2.1.6 of IEEE Std 1722.1-2013).

Copyright © 2016 IEEE. All rights reserved.

Annex I

(normative)

AVDECC stream formats

I.1 Overview

Audio/Video Discovery, Enumeration, Connection Management, and Control (AVDECC) defines a mechanism for describing the contents of a Stream such that an AVDECC Controller can determine what formats an AVDECC Talker and AVDECC Listener support, what they have in common, and hence, what can be used to transfer data between the two. This mechanism is based on an 8-octet value known as a *Stream Format*. This annex defines the standard stream formats for time-sensitive streams as defined by this standard. This annex supersedes subclause 7.3.2.1 of IEEE Std 1722.1-2013.

I.2 Stream formats

A stream format is defined in subclause 7.3.2 of IEEE Std 1722.1-2013. This annex applies when the **v** field has the value zero (0).

I.2.1 Standard Stream Format

A stream format with the **v** field set to zero (0) describes an AVTP defined time-sensitive stream payload.

Figure I.1 describes the basic binary stream format that contains the following fields:

— **v** (version) field: 1 bit

— **subtype** field: 7 bits

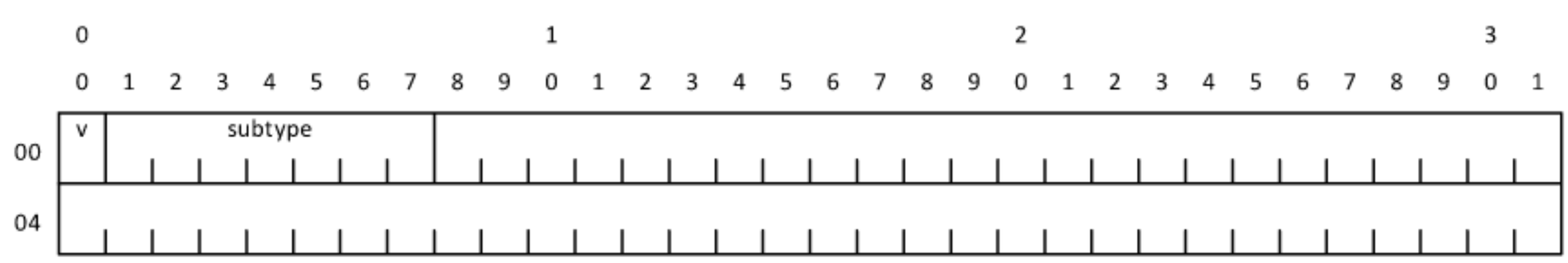


Figure I.1—Standard Stream Format

The **subtype** field is the lower 7-bits of the stream's subtype as defined in 4.4.3.2. As such, stream formats are limited to the first 127 subtypes only.

The meanings of the remaining bits in the stream format are determined by the **subtype** field's value.

Copyright © 2016 IEEE. All rights reserved.

I.2.2 IEC 61883/IIDC Stream Format

I.2.2.1 Overview

A stream format with the **subtype** set to 61883_IIDC represents an IEC 61883 or IIDC format stream.

Figure I.2 defines the stream format used for IEC 61883/IIDC streams that are conforming to one of the following standards:

— IEC 61883-4: MPEG2-TS data transmission

— IEC 61883-6: Audio and music data transmission protocol

— IEC 61883-8: Transmission of ITU-R BT.601–style digital video data

— IIDC2 Digital Camera Control Specification Ver. 1.0.0, Jan. 2012

The 61883_IIDC_SUBTYPE Stream Format adds the following field:

— **sf** field: 1 bit

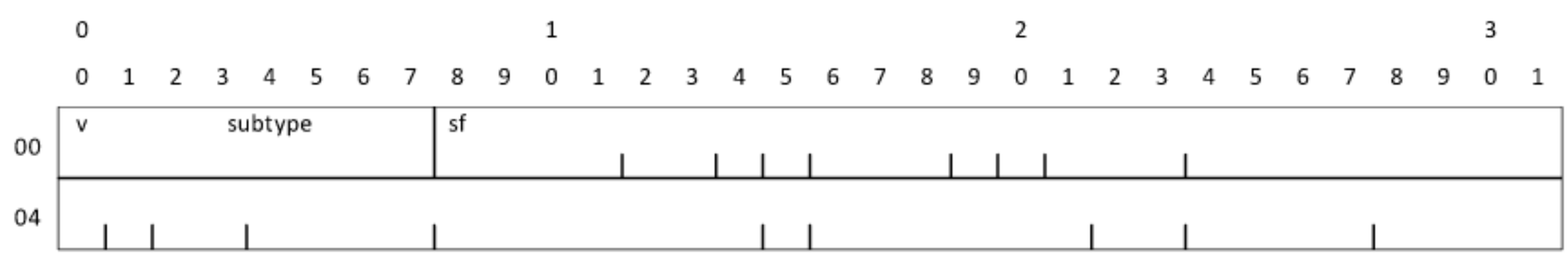


Figure I.2—IEC 61883/IIDC Stream Format

The **sf** field may be:

— 0: The stream format describes an IIDC format stream

— 1: The stream format describes an IEC 61883-4, IEC 61883-6, or IEC 61883-8 format stream

I.2.2.2 IIDC Stream Format

Figure I.3 defines the stream format used for IIDC format streams. It adds the following fields:

— **iidc_format** field: 2 octets

— **iidc_rate** field: 1 octet

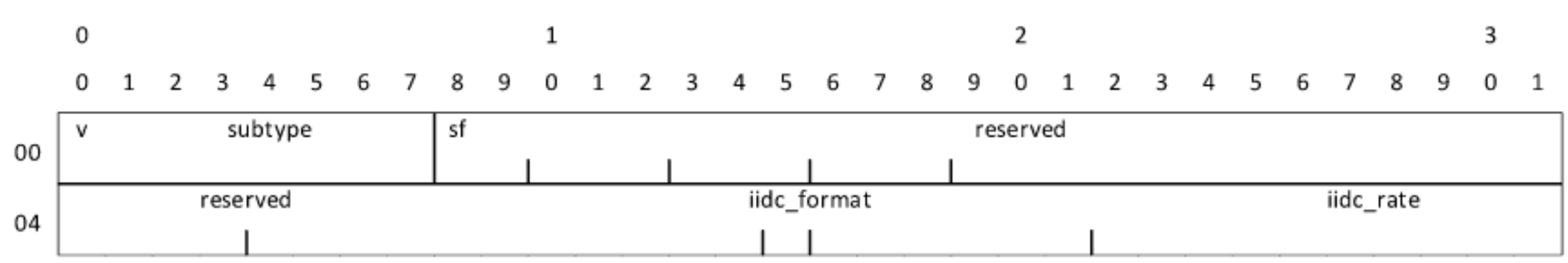


Figure I.3—IIDC Stream Format

Copyright © 2016 IEEE. All rights reserved.

The **iidc_format** field identifies the pixel encoding used in the IIDC stream. The values are defined in Table I.1.

Table I.1—IIDC Pixel Encoding Format

Value	Name	Description
0	VGA_MONO8	VGA (640 × 480) 8-bit monochrome
1	VGA_RGB8_PACKED	VGA (640 × 480) 8-bit RGB packed in 3 octets
2	VGA_YUV422_PACKED	VGA (640 × 480) YUV 4:2:2 packed in 2 octets
3 to 7	—	Reserved
8	XGA_MONO8	XGA (1024 × 768) 8-bit monochrome
9	XGA_RGB8_PACKED	XGA (1024 × 768) 8-bit RGB packed in 3 octets
10	XGA_YUV422_PACKED	XGA (1024 × 768) YUV 4:2:2 packed in 2 octets
11 to 15	—	Reserved
16	HD720P_MONO8	HD720p (1280 × 720) YUV 4:2:2 packed in 2 octets
17	HD720P_RGB8_PACKED	HD720p (1280 × 720) 8-bit RGB packed in 3 octets
18	HD720P_YUV422_PACKED	HD720p (1280 × 720) YUV 4:2:2 packed in 2 octets
19 to 23	—	Reserved
24	SXGA_MONO8	SXGA (1280 × 960) 8-bit monochrome
25	SXGA_RGB8_PACKED	SXGA (1280 × 960) 8-bit RGB packed in 3 octets
26	SXGA_YUV422_PACKED	SXGA (1280 × 960) YUV 4:2:2 packed in 2 octets
27 to 31	—	Reserved
32	UXGA_MONO8	UXGA (1600 × 1200) 8-bit monochrome
33	UXGA_RGB8_PACKED	UXGA (1600 × 1200) 8-bit RGB packed in 3 octets
34	UXGA_YUV422_PACKED	UXGA (1600 × 1200) YUV 4:2:2 packed in 2 octets
35 to 39	—	Reserved
40	HD1080P_MONO8	HD1080p (1920 × 1080) 8-bit monochrome
41	HD1080P_RGB8_PACKED	HD1080p (1920 × 1080) 8-bit RGB packed in 3 octets
42	HD1080P_YUV422_PACKED	HD1080p (1920 × 1080) YUV 4:2:2 packed in 2 octets
43 to 255	—	Reserved
256	HVGA_MONO8	HVGA (320 × 240) 8-bit monochrome
257	HVGA_RGB8_PACKED	HVGA (320 × 240) 8-bit RGB packed in 3 octets
258	HVGA_YUV422_PACKED	HVGA (320 × 240) YUV 4:2:2 packed in 2 octets
259 to 4095	—	Reserved
4096	VGA_ARGB32	VGA (640 × 480) 8-bit ARGB packet into 4 octets
4097	XGA_ARGB32	XGA (1024 × 768) 8-bit ARGB packet into 4 octets
4098	HD720P_ARGB32	HD720p (1280 × 720) 8-bit ARGB packet into 4 octets
4099	SXGA_ARGB32	SXGA (1280 × 960) 8-bit ARGB packet into 4 octets
4100	UXGA_ARGB32	UXGA (1600 × 1200) 8-bit ARGB packet into 4 octets
4101	HD1080P_ARGB32	HD1080p (1920 × 1080) 8-bit ARGB packet into 4 octets
4102	HVGA_ARGB32	HVGA (320 × 240) 8-bit ARGB packet into 4 octets
4103 to 4351	—	Reserved
4352	VGA_BGRA32	VGA (640 × 480) 8-bit BGRA packet into 4 octets
4353	XGA_BGRA32	XGA (1024 × 768) 8-bit BGRA packet into 4 octets
4354	HD720P_BGRA32	HD720p (1280 × 720) 8-bit BGRA packet into 4 octets
4355	SXGA_BGRA32	SXGA (1280 × 960) 8-bit BGRA packet into 4 octets
4356	UXGA_BGRA32	UXGA (1600 × 1200) 8-bit BGRA packet into 4 octets
4357	HD1080P_BGRA32	HD1080p (1920 × 1080) 8-bit BGRA packet into 4 octets
4358	HVGA_BGRA32	HVGA (320 × 240) 8-bit BGRA packet into 4 octets
4359 to 65 534	—	Reserved
65 535	CUSTOM	Custom format defined by Video Cluster

The **iidc_rate** field identifies the video frame rate of the IIDC stream. The values are defined in Table I.2.

Copyright © 2016 IEEE. All rights reserved.

Table I.2—IIDC video frame rates

Value	Name	Description
0	RATE_1_875	1.875 video frames per second
1	RATE_3_75	3.75 video frames per second
2	RATE_7_5	7.5 video frames per second
3	RATE_15	15 video frames per second
4	RATE_30	30 video frames per second
5	RATE_60	60 video frames per second
6	RATE_120	120 video frames per second
7	RATE_240	240 video frames per second
8 to 127	—	Reserved
128	RATE_1	1 video frame per second
129	RATE_2	2 video frames per second
130	RATE_3	3 video frames per second
131	RATE_4	4 video frames per second
132	RATE_5	5 video frames per second
133	RATE_6	6 video frames per second
134	RATE_8	8 video frames per second
135	RATE_10	10 video frames per second
136	RATE_12	12 video frames per second
137	RATE_16	16 video frames per second
138	RATE_20	20 video frames per second
139	RATE_24	24 video frames per second
140	RATE_25	25 video frames per second
141	RATE_32	32 video frames per second
142	RATE_40	40 video frames per second
143	RATE_50	50 video frames per second
144	RATE_100	100 video frames per second
145	RATE_150	150 video frames per second
146	RATE_200	200 video frames per second
147	RATE_15_PAL_PULLDOWN	15 × 24/25 video frames per second
148	RATE_24_PAL_PULLDOWN	24 × 24/25 video frames per second
149	RATE_25_PAL_PULLDOWN	25 × 24/25 video frames per second
150	RATE_30_PAL_PULLDOWN	30 × 24/25 video frames per second
151	RATE_50_PAL_PULLDOWN	50 × 24/25 video frames per second
152	RATE_60_PAL_PULLDOWN	60 × 24/25 video frames per second
153	RATE_120_PAL_PULLDOWN	120 × 24/25 video frames per second
154	RATE_240_PAL_PULLDOWN	240 × 24/25 video frames per second
155	RATE_15_PAL_PULLUP	15 × 25/24 video frames per second
156	RATE_24_PAL_PULLUP	24 × 25/24 video frames per second
157	RATE_25_PAL_PULLUP	25 × 25/24 video frames per second
158	RATE_30_PAL_PULLUP	30 × 25/24 video frames per second
159	RATE_50_PAL_PULLUP	50 × 25/24 video frames per second
160	RATE_60_PAL_PULLUP	60 × 25/24 video frames per second
161	RATE_120_PAL_PULLUP	120 × 25/24 video frames per second
162	RATE_240_PAL_PULLUP	240 × 25/24 video frames per second
163	RATE_15_NTSC_PULLDOWN	15 × 1000/1001 video frames per second
164	RATE_24_NTSC_PULLDOWN	24 × 1000/1001 video frames per second
165	RATE_25_NTSC_PULLDOWN	25 × 1000/1001 video frames per second
166	RATE_30_NTSC_PULLDOWN	30 × 1000/1001 video frames per second

Copyright © 2016 IEEE. All rights reserved.

Table I.2—IIDC video frame rates *(continued)*

167	RATE_50_NTSC_PULLDOWN	50 × 1000/1001 video frames per second
168	RATE_60_NTSC_PULLDOWN	60 × 1000/1001 video frames per second
169	RATE_120_NTSC_PULLDOWN	120 × 1000/1001 video frames per second
170	RATE_240_NTSC_PULLDOWN	240 × 1000/1001 video frames per second
171	RATE_15_NTSC_PULLUP	15 × 1001/1000 video frames per second
172	RATE_24_NTSC_PULLUP	24 × 1001/1000 video frames per second
173	RATE_25_NTSC_PULLUP	25 × 1001/1000 video frames per second
174	RATE_30_NTSC_PULLUP	30 × 1001/1000 video frames per second
175	RATE_50_NTSC_PULLUP	50 × 1001/1000 video frames per second
176	RATE_60_NTSC_PULLUP	60 × 1001/1000 video frames per second
177	RATE_120_NTSC_PULLUP	120 × 1001/1000 video frames per second
178	RATE_240_NTSC_PULLUP	240 × 1001/1000 video frames per second
179 to 254	—	Reserved
255	CUSTOM	Custom video frame rate defined by Video Cluster

I.2.2.3 IEC 61883 Stream Format

I.2.2.3.1 Overview

Figure I.4 defines the stream format used for IEC 61883–based format streams. It adds the following field, which directly represents the associated field in the IEC 61883 stream header:

— **fmt** field: 6 bits

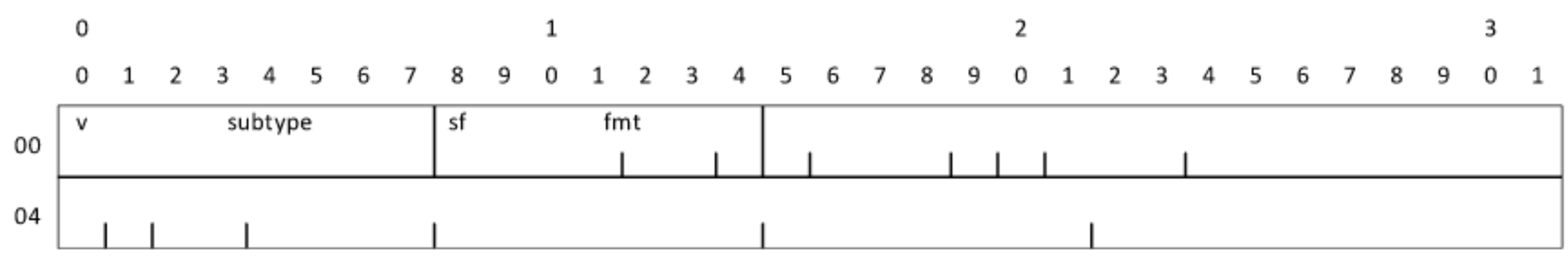


Figure I.4—IEC 61883 Stream Format

The **fmt** field represents the FMT field defined in IEC 61883-1:2008 Clause 6.2.1. The usage of the **fmt** field is defined in Table I.3.

Table I.3—IEC 61883 Format IDs

Value	Name	Description
20_{16}	IEC_61883_4_FMT	IEC 61883-4 MPEG2 TS
10_{16}	IEC_61883_6_FMT	IEC 61883-6 Audio/Music
21_{16}	IEC_61883_7_FMT	IEC 61883-7 ITU-R BO.1294 System B
01_{16}	IEC_61883_8_FMT	IEC 61883-8 BT.601 Video

Copyright © 2016 IEEE. All rights reserved.

I.2.2.3.2 IEC 61883-4 Stream Format

Figure I.5 defines the stream format used for IEC 61883-4 format streams. Since IEC 61883-4 format streams do not add any additional information from the stream header, the IEC 61883-4 Stream Format adds a single reserved field:

— **reserved** field: 49 bits

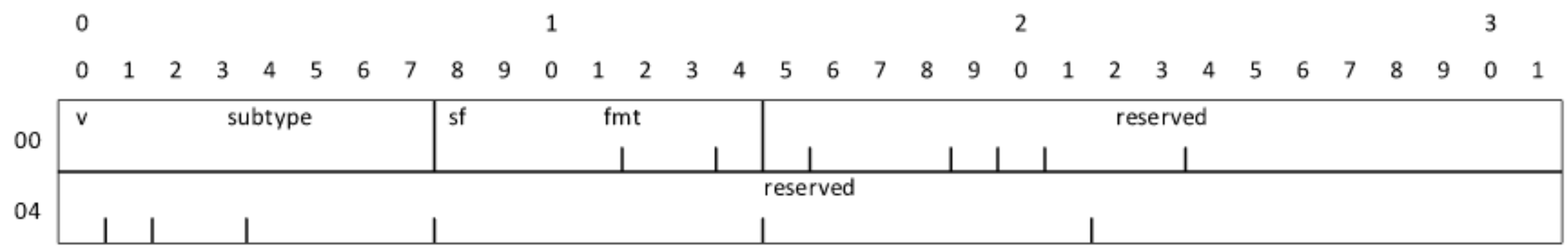


Figure I.5—IEC 61883-4 Stream Format

I.2.2.3.3 IEC 61883-6 Stream Format

Figure I.6 defines the general stream format used for IEC 61883-6 format streams. It adds the following fields that directly represent the associated fields in the IEC 61883-6 stream header:

— **fdf_evt**: 5 bits

— **fdf_sfc**: 3 bits

— **dbs**: 1 octet

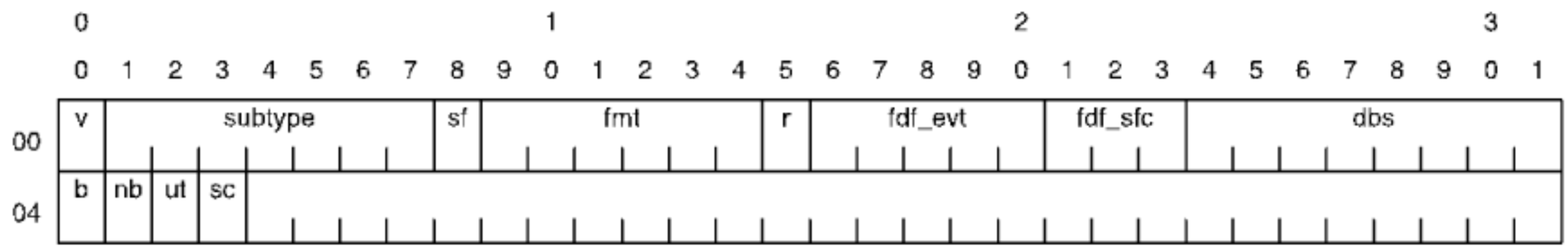


Figure I.6—IEC 61883-6 Stream Format

The **fdf_evt** field represents the most-significant 5 bits of the FDF field defined in IEC 61883-6:2014 Clause 9.1.

The **f2f_sfc** field represents the least-significant 3 bits of the FDF field defined in IEC 61883-6:2014 Clause 9.1.

The **dbs** field represents the DBS field defined in IEC 61883-6:2014 Clause 9.2.

The IEC 61883-6 stream format also includes the following fields:

— **b**: 1 bit

— **nb**: 1 bit

— **ut**: 1 bit

— **sc**: 1 bit

Copyright © 2016 IEEE. All rights reserved.

The **b** field is used to determine if an IEC 61883-6–capable stream supports blocking mode as defined in IEC 61883-6:2014 Annex A. The field shall be set to one (1) if the stream supports blocking mode, or zero (0) otherwise.

The **nb** field is used to determine if an IEC 61883-6–capable stream supports non-blocking mode as defined in IEC 61883-6:2014 Annex A. The field shall be set to one (1) if the stream supports non-blocking mode, or zero (0) otherwise.

While the **b** and **nb** fields may both be set to one (1) in a stream format used in the list of supported formats, only one of them shall be set to one (1) when the stream format is the current format of the stream and in a SET_STREAM_FORMAT command.

The **ut** field is used to determine if the IEC 61883-6 stream source or sink is capable of sourcing or sinking a stream with less than the number of data blocks indicated by the **dbs** field. When set to one (1), the stream source or sink is capable of using fewer data blocks than specified. The **ut** field shall be set to zero (0) when the stream format is the current format of the stream and when used in a SET_STREAM_FORMAT command.

The **sc** field is used to indicate that the stream uses a packetization clock that is synchronous to the media clock. When set to one (1) the packetization clock and media clock are synchronous (e.g., a 48-kHz stream is always sending 6 samples per PDU), when set to zero (0) the packetization clock and media clock are asynchronous.

NOTE—AVDECC entities utilizing the stream formats as published in IEEE Std 1722.1 are always considered asynchronous.

The usages of the remaining bits depend on the value of the **fdf_evt** field:

— 00110_2: IEC 61883-6 32-bit fixed-point packetization

— 00100_2: IEC 61883-6 32-bit floating-point packetization

— 00000_2: IEC 61883-6 AM824 packetization

I.2.2.3.4 IEC 61883-6 32-bit Stream Format

Figure I.7 is used to describe a stream of IEC 61883-6 32-bit fixed-point multichannel audio data as defined in IEC 61883-6:2014 Clause 8.5.

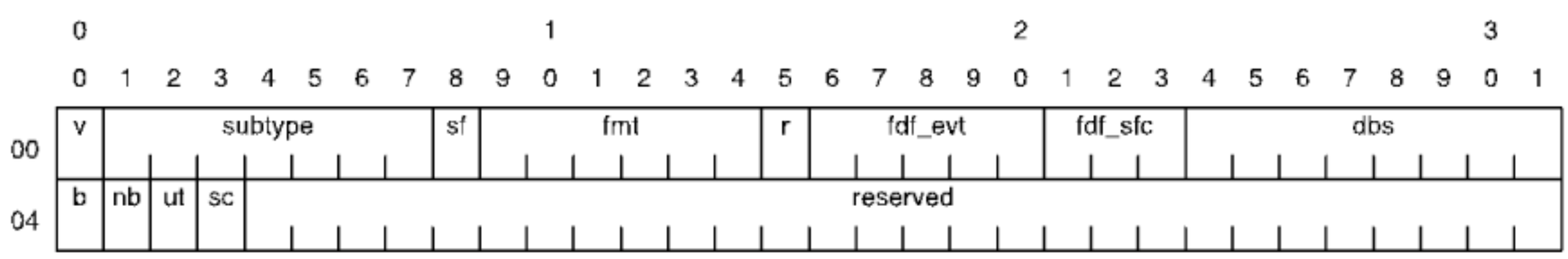


Figure I.7—IEC 61883-6 32-bit Stream Format

I.2.2.3.5 IEC 61883-6 Float Stream Format

Figure I.8 is used to describe a stream of IEC 61883-6 32-bit floating-point multichannel audio data as defined in IEC 61883-6:2014 Clause 8.3.

Copyright © 2016 IEEE. All rights reserved.

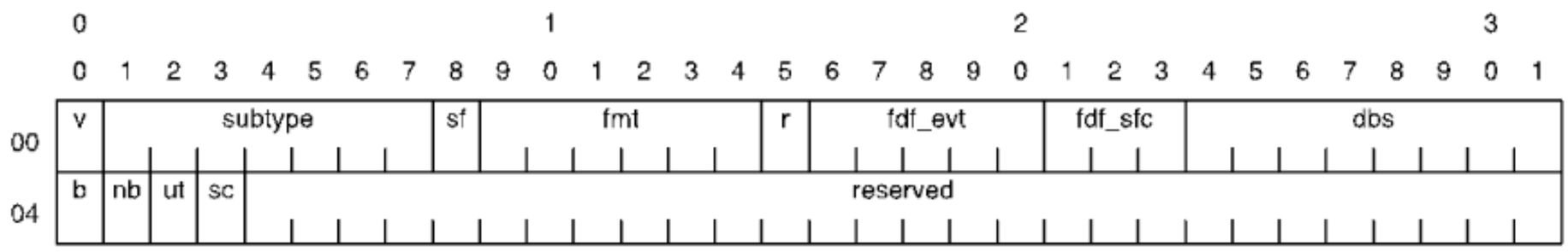


Figure I.8—IEC 61883-6 Float Stream Format

I.2.2.3.6 IEC 61883-6 AM824 Stream Format

Figure I.9 is used to describe a stream in IEC 61883-6 AM824 format as defined in IEC 61883-6:2014 Clause 8.2. This stream format adds the following fields:

— **label_iec_60958_cnt**: 1 octet, count of iec60958 quadlets. See IEC 61883-6:2014 Clause 8.2.2

— **label_mbla_cnt**: 1 octet, count of multi-bit linear audio quadlets. See IEC 61883-6:2014 Clause 8.2.3

— **label_midi_cnt**: 4 bits, count of MIDI quadlets. See IEC 61883-6:2014 Clause 8.2.5

— **label_smptecnt**: 4 bits, count of SMPTE quadlets. See IEC 61883-6:2014 Clause 8.2.6

The sum of these four fields is required to be equal to the value of the **dbs** field.

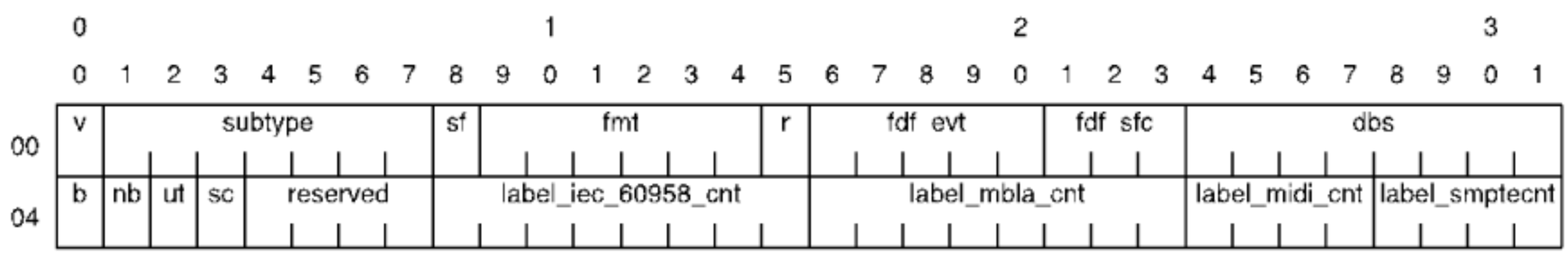


Figure I.9—IEC 61883-6 AM824 Stream Format

I.2.2.3.7 IEC 61883-7 Stream Format

Figure I.10 defines the stream format used for IEC 61883-7 format streams. Since IEC 61883-7 format streams do not add any additional information from the stream header the IEC 61883-7 Stream Format adds a single reserved field:

— **reserved** field: 49 bits

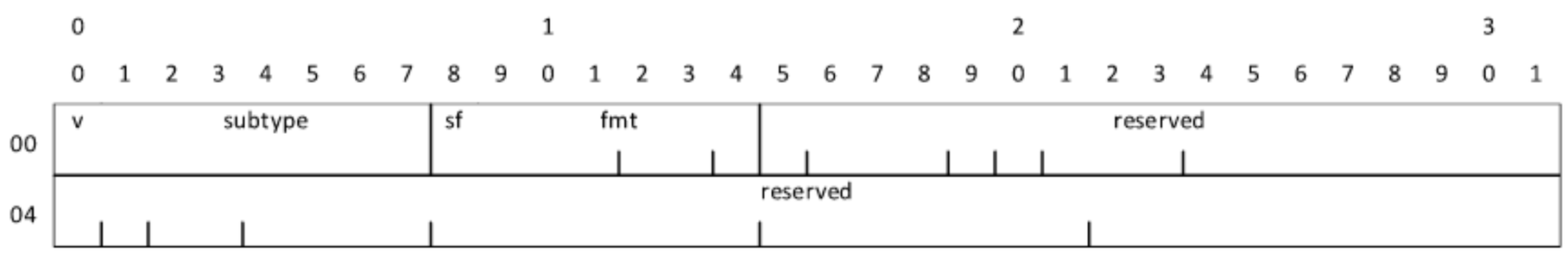


Figure I.10—IEC 61883-7 Stream Format

Copyright © 2016 IEEE. All rights reserved.

I.2.2.3.8 IEC 61883-8 Stream Format

Figure I.11 describes a stream in the IEC 61883-8 format. It adds the following fields:

— **video_mode**: 1 octet

— **compress_mode**: 1 octet

— **color_space**: 1 octet

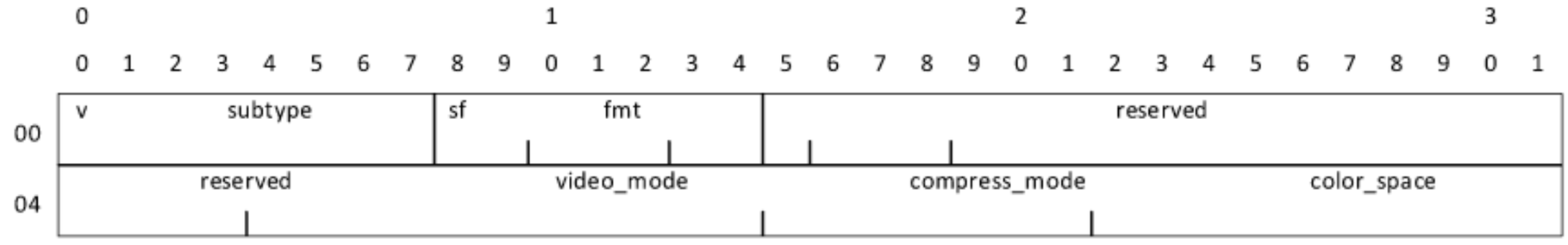


Figure I.11—IEC 61883-8 Stream Format

I.2.3 MMA Stream Format

A stream format with the **subtype** set to MMA_STREAM represents an MMA format stream.

The MMA format stream format is defined by the MMA. MMA stands for the Musical Instrument Digital Interface (MIDI) Manufacturers Association, which is the publisher and authoritative source of MIDI specifications. The MMA publishes the MMA Payload Format Specification for AVTP [B24][1], which defines the transport of MMA defined data over AVTP.

I.2.4 AVTP Audio Stream Format

A stream format with the **subtype** set to AAF represents an AVTP Audio Format stream.

Figure I.12 defines the stream format used for AVTP Audio Format streams. It adds the following fields:

— **ut**: 1 bit

— **nsr**: 4 bits

— **format**: 1 octet

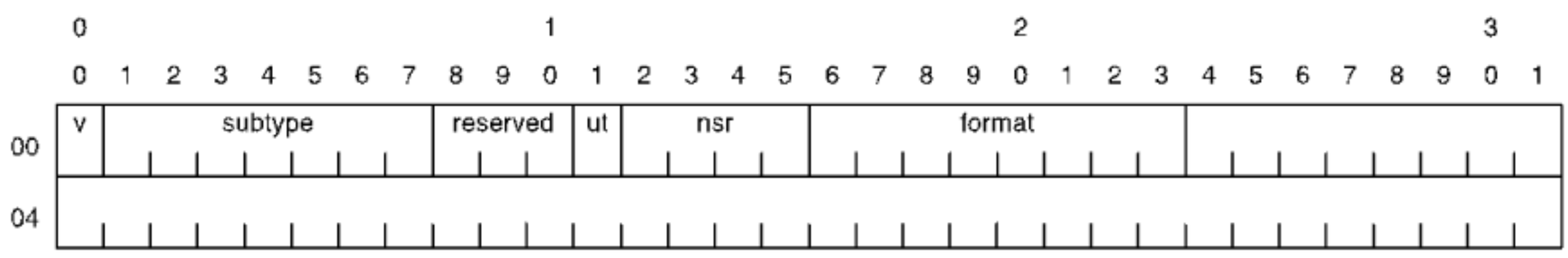


Figure I.12—AVTP Audio Format Stream Format

The **ut** field is used to determine if the AAF stream source or sink is capable of sourcing or sinking a stream with less than the number of channels indicated by the **channels_per_frame** field. When set to one (1) the stream source or sink is capable of using fewer channels than specified. The **ut** field shall be set to

[1]The MIDI Manufacturers Association Inc., La Habra, CA, USA; http://www.midi.org/avbtp.

Copyright © 2016 IEEE. All rights reserved.

zero (0) when the stream format is the current format of the stream and when used in a SET_STREAM_FORMAT command.

The **nsr** field contains the nominal base frequency of the stream. It is set to one of the values defined in Table 11. This field matches the setting of the **nsr** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **format** field describes the type of samples contained within the stream. It is set to one of the values defined in Table 9. This field matches the setting of the **format** field in the Stream AVTPDU when this stream format is the current format for the stream.

I.2.4.1 PCM Stream Format

Figure I.13 defines the stream format used for AAF PCM format streams. It adds the following fields:

— **bit_depth**: 1 octet

— **channels_per_frame**: 10 bits

— **samples_per_frame**: 10 bits

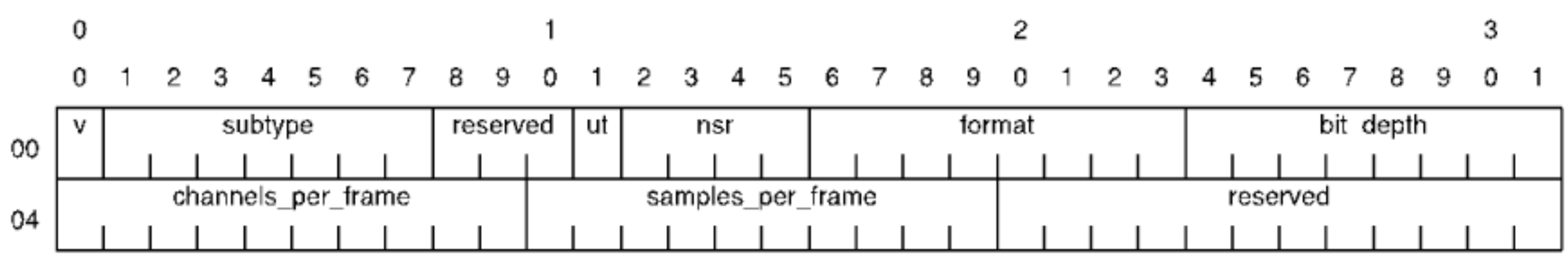


Figure I.13—PCM AVTP Audio Format Stream Format

The **bit_depth** field describes how many bits within the sample are actually used for data. The value of **bit_depth** is not to be set to a number that is larger than the size of the format set in the **format** field. If the **format** field is set to a non-integer format, then the **bit_depth** field shall be set to zero (0). This field matches the setting of the **bit_depth** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **channels_per_frame** field describes how many channels of samples are contained within each frame of the stream. This field matches the setting of the **channels_per_frame** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **samples_per_frame** field describes how many samples for each channel are contained within each frame of the stream.

I.2.4.2 AES3 Stream Format

Figure I.14 defines the stream format used for AAF AES3 format streams. It adds the following fields:

— **frames_per_frame** field: 1 octet

— **streams_per_frame** field: 10 bits

— **aes3_dt_ref** field: 3 bits

— **aes3_data_type** field: 2 octets

Copyright © 2016 IEEE. All rights reserved.

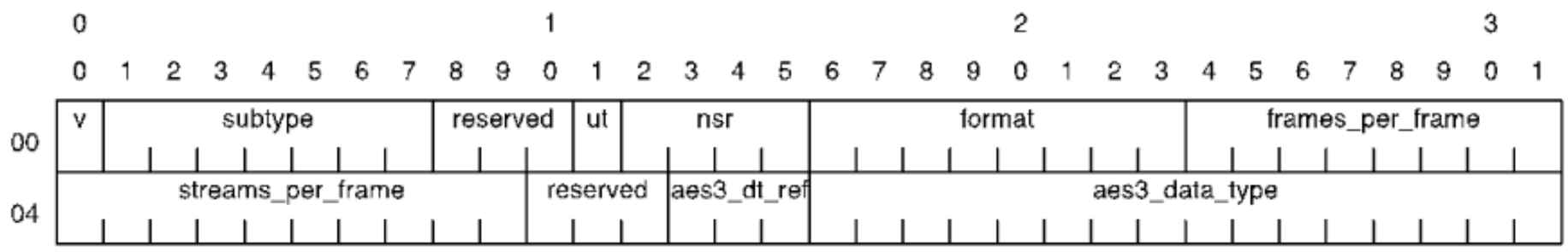


Figure I.14—AES3 AVTP Audio Format Stream Format

The **frames_per_frame** field describes how many AES3 frames for each channel are contained within each AVTPDU of the stream.

The **streams_per_frame** field describes how many AES3 streams are contained within each frame of the stream. This field matches the setting of the **streams_per_frame** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **aes3_dt_ref** field describes the type of data types used by the stream. This field matches the setting of the **aes3_dt_ref** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **aes3_data_type** field describes the type of data that is contained within the AES3 frames of the stream. This field matches the setting of the **aes3_data_type** field (**aes3_data_type_l** and **aes3_data_type_h**) in the Stream AVTPDU when this stream format is the current format for the stream.

I.2.5 Compressed Video Stream Format

I.2.5.1 Common CVF Stream Format

A stream format with the **subtype** set to CVF represents a Compressed Video Format stream.

Figure I.15 defines the stream format used for Compressed Video Format streams. It adds the following fields:

— **format** field: 1 octet

— **format_subtype** field: 1 octet

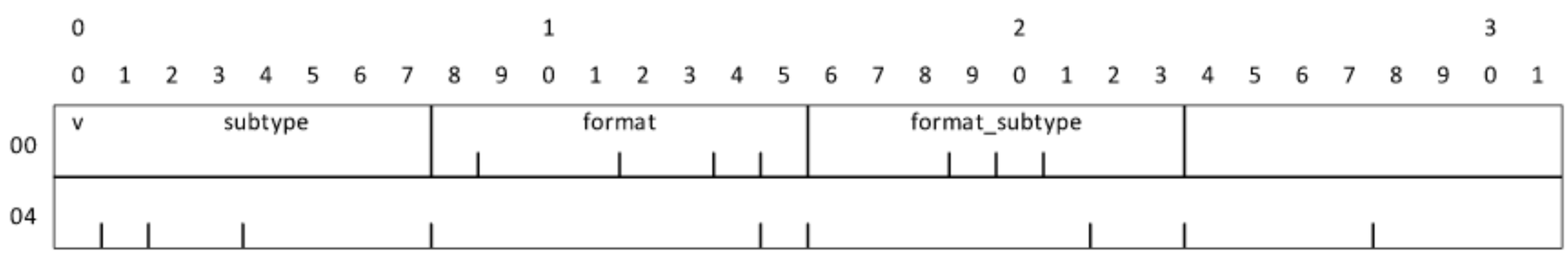


Figure I.15—Compressed Video Format Stream Format

The **format** field describes the family of compression used by the stream. It is set to one of the values defined in Table 19. This field matches the setting of the **format** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **format_subtype** field describes the type of compression used by the stream. It is set to one of the values defined in Table 20. This field matches the setting of the **format_subtype** field in the Stream AVTPDU when this stream format is the current format for the stream.

Copyright © 2016 IEEE. All rights reserved.

I.2.5.2 MJPEG Stream Format

Figure I.13 defines the stream format used for MJPEG Compressed Video Format streams. It adds the following fields:

— **p** field: 1 bit

— **type** field: 1 octet

— **width** field: 1 octet

— **height** field: 1 octet

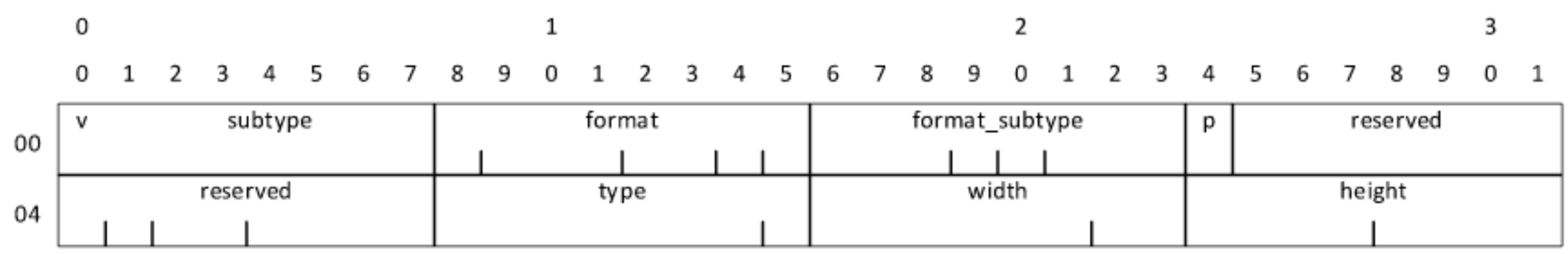


Figure I.16—MJPEG Compressed Video Format Stream Format

The **p** field indicates if the stream is using progressive (**type_specific** of zero [0]) scan or interlace (**type_specific** of 1, 2, or 3) scanning.

The **type** field describes the type of video being encoded in the stream. This field matches the **type** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **width** field describes the width of the video being encoded in the stream in 8-pixel multiples. This field matches the **width** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **height** field describes the height of the video being encoded in the stream in 8-pixel multiples. This field matches the **height** field in the Stream AVTPDU when this stream format is the current format for the stream.

I.2.5.3 H.264 Stream Format

Figure I.14 defines the stream format used for H.264 Compressed Video Format streams. It does not add any fields.

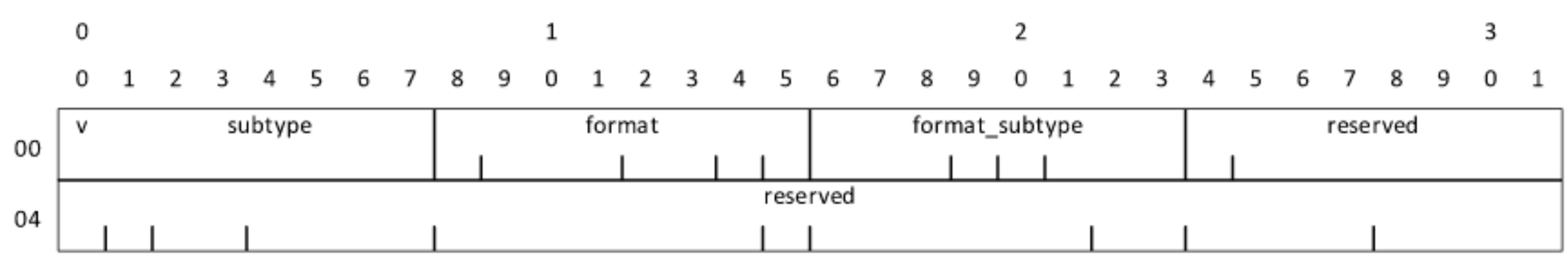


Figure I.17—H.264 Compressed Video Format Stream Format

Copyright © 2016 IEEE. All rights reserved.

I.2.5.4 JPEG2000 Stream Format

Figure I.15 defines the stream format used for JPEG2000 Compressed Video Format streams. It adds the following fields:

— **p** field: 1 bit

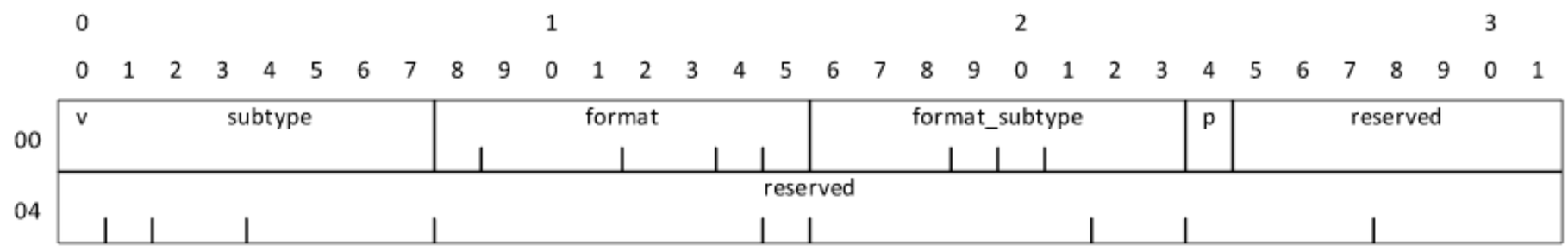


Figure I.18—JPEG2000 Compressed Video Format Stream Format

The **p** field indicates if the stream is using progressive (**tp** of zero [0]) scan or interlace (**tp** of 1 or 2) scanning.

I.2.6 Clock Reference Stream Format

A stream format with the **subtype** set to CRF represents a Clock Reference Format stream.

Figure I.16 defines the stream format used for Clock Reference Format streams. It adds the following fields:

— **type** field: 4 bits

— **timestamp_interval** field: 12 bits

— **timestamps_per_pdu** field: 1 octet

— **pull** field: 3 bits

— **base_frequency** field: 12 bits

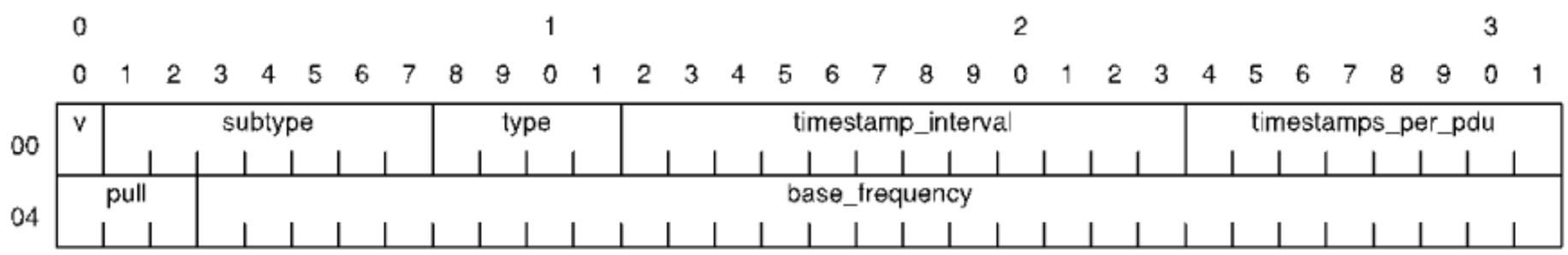


Figure I.19—Clock Reference Format Stream Format

The **type** field describes the type of clock carried by the stream. It is set to one of the values defined in Table 26. This field matches the setting of the **type** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **timestamp_interval** field describes the number of events (audio samples, video frames, video lines, etc.) between the timestamps provided by the stream. This field matches the setting of the **timestamp_interval** field in the Stream AVTPDU when this stream format is the current format for the stream.

Copyright © 2016 IEEE. All rights reserved.

The **timestamps_per_pdu** field describes the number of timestamps that are contained within each CRF AVTPDU within the stream.

The **pull** field describes the multiplier applied to the **base_frequency** field to calculate the nominal frequency of the clock carried by the stream. This field matches the **pull** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **base_frequency** field describes the frequency in Hertz before being multiplied to calculate the nominal frequency of the clock carried by the stream. This field matches the **base_frequency** field in the Stream AVTPDU when this stream format is the current format for the stream.

I.2.7 Time-Synchronous Control Stream Format

A stream format with the **subtype** set to TSCF represents a Time-Sensitive Control Format stream.

Figure I.17 defines the stream format used for Time-Sensitive Control streams. It adds the following fields:

— **m** field: 1 bit
— **t3v** field: 1 bit
— **type_3** field: 7 bits
— **t2v** field: 1 bit
— **type_2** field: 7 bits
— **t1v** field: 1 bit
— **type_1** field: 7 bits
— **t0v** field: 1 bit
— **type_0** field: 7 bits

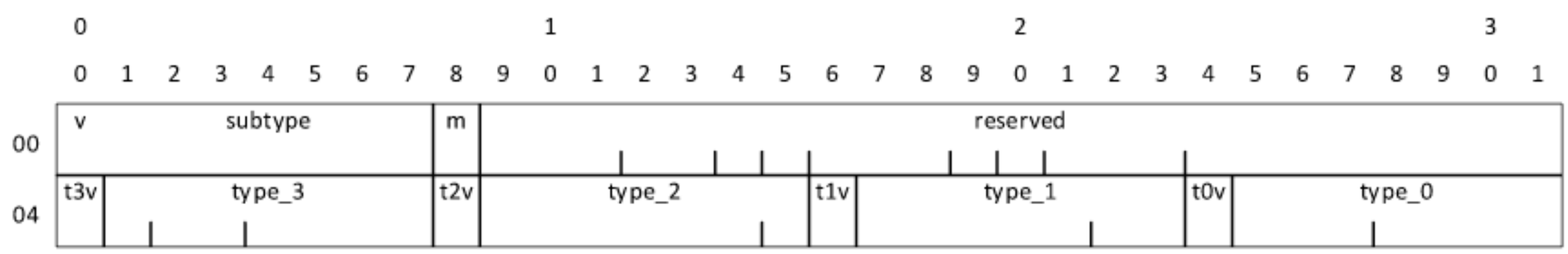


Figure I.20—Time-Sensitive Control Format Stream Format

The **m** field indicates if the stream carries more than 4 types of messages. It is set to one (1) if the stream has more than 4 message types, zero (0) otherwise.

The **t3v** field indicates if the **type_3** field contains a message type. It is set to one (1) if the field contains a value or zero (0) otherwise.

The **type_3** field indicates a message type that is carried by the stream. It is set to one of the values from Table 22 if the stream carries four or more message types.

The **t2v** field indicates if the **type_2** field contains a message type. It is set to one (1) if the field contains a value or zero (0) otherwise.

Copyright © 2016 IEEE. All rights reserved.

The **type_2** field indicates a message type that is carried by the stream. It is set to one of the values from Table 22 if the stream carries three or more message types.

The **t1v** field indicates if the **type_1** field contains a message type. It is set to one (1) if the field contains a value or zero (0) otherwise.

The **type_1** field indicates a message type that is carried by the stream. It is set to one of the values from Table 22 if the stream carries two or more message types.

The **t0v** field indicates if the **type_0** field contains a message type. It is set to one (1) if the field contains a value or zero (0) otherwise.

The **type_0** field indicates a message type that is carried by the stream. It is set to one of the values from Table 22 if the stream carries one or more message types.

I.2.8 SDI Video Stream Format

A stream format with the **subtype** set to SVF represents a Serial Digital Interface (SDI) Video Format stream.

Figure I.18 defines the stream format used for SDI Video Format streams. It adds the following fields:

— **format** field: 1 octet

— **map** field: 4 bits

— **sample** field: 4 bits

— **frame** field: 1 octet

— **frate** field: 1 octet

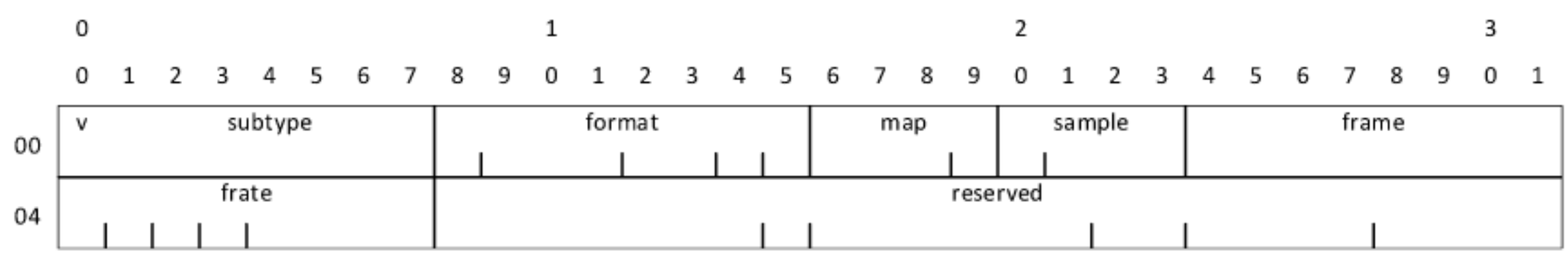


Figure I.21—SDI Video Format Stream Format

The **format** field describes the format of the video contained by the stream. It is set to one of the values defined in Table 32. This field matches the setting of the **format** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **map** field describes the top-level data structure of the data carried by the stream. It is set to one of the values defined in Table 35. This field matches the setting of the **map** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **sample** field describes the pixel-sampling structure and bit depth that is carried by the stream. It is set to one of the values defined in Table 36. This field matches the setting of the **sample** field in the Stream AVTPDU when this stream format is the current format for the stream.

Copyright © 2016 IEEE. All rights reserved.

The **frame** field describes the active pixel structure of the data carried by the stream. It is set to one of the values defined in Table 37. This field matches the setting of the **frame** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **frate** field describes the top-level data structure of the data carried by the stream. It is set to one of the values defined in Table 38. This field matches the setting of the **frate** field in the Stream AVTPDU when this stream format is the current format for the stream.

I.2.9 Raw Video Stream Format

A stream format with the **subtype** set to RVF represents a Raw Video Format stream.

Figure I.22 defines the stream format used for Raw Video Format streams. It adds the following fields:

— **pd** field: 1 bit
— **i** field: 1 bit
— **colorspace** field: 4 bits
— **pixel_depth** field: 4 bits
— **pixel_format** field: 4 bits
— **frame_rate** field: 1 octet
— **active_pixels** field: 2 octets
— **total_lines** field: 2 octets

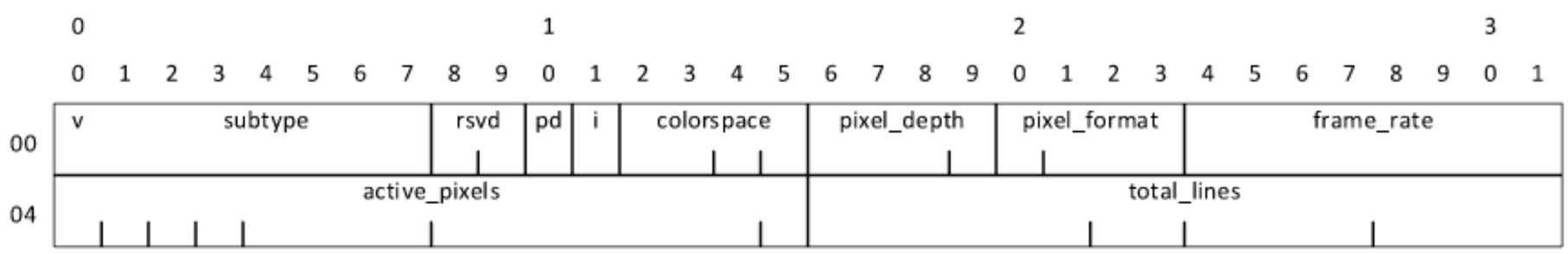


Figure I.22—Raw Video Format Stream Format

The **pd** field describes the pull-down multipier of the rate of the video contained by the stream. When set to one (1) the rate identified by the **frame_rate** field is multiplied by 1000/1001, otherwise it is used directly. This field matches the setting of the **pd** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **i** field describes the interlacing of the video contained by the stream. When set to one (1) the payload data is interlaced otherwise the payload data is non-interlaced (progressive). This field matches the setting of the **i** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **colorspace** field describes the color space of the video contained by the stream. It is set to one of the values defined in Table 43. This field matches the setting of the **colorspace** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **pixel_depth** field describes the number of bits allocated to each pixel of the video contained by the stream. It is set to one of the values defined in Table 40. This field matches the setting of the **pixel_depth** field in the Stream AVTPDU when this stream format is the current format for the stream.

Copyright © 2016 IEEE. All rights reserved.

The **pixel_format** field describes the pixel sampling structure of the data carried by the stream. It is set to one of the values defined in Table 41. This field matches the setting of the **pixel_format** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **frame_rate** field describes the pixel sampling structure and bit depth that is carried by the stream. It is set to one of the values defined in Table 42. This field matches the setting of the **frame_rate** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **active_pixels** field describes the number of active pixels in each line of video carried by the stream. This field matches the setting of the **active_pixel** field in the Stream AVTPDU when this stream format is the current format for the stream.

The **total_lines** field describes the total number of lines contained within a video frame carried by the stream. This field matches the setting of the **total_lines** field in the Stream AVTPDU when this stream format is the current format for the stream.

I.2.10 AES Encrypted Stream Format

The AES Encrypted Format is an encryption encapsulation layer around a media stream. Since the media stream is the content that is required to be compatible between the Talker and Listener it is the encapsulated media stream's stream format that is used to identify the content of the stream. As such there is no Stream Format defined with the **subtype** set to AEF_CONTINUOUS, however this is reserved for future use.

I.2.11 Vendor Specific Stream Format

A stream format with the **subtype** set to VSF_STREAM represents a vendor defined AVTP streaming format.

Figure I.20 defines the stream format for vendor-specific streaming formats. It adds the following fields:

— **format_id**: 48 bits, the vendor's ID value identifying the streaming format. This is the same value as used in the AVTPDU's vendor_id_1 and vendor_id_2 fields (14.1.1.2).

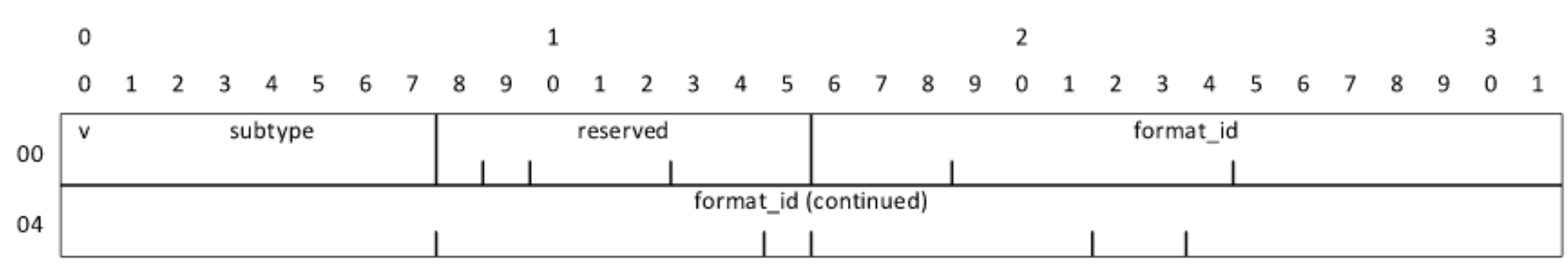


Figure I.23—Vendor Specific Stream Format

I.2.12 Experimental Stream Format

A stream format with the **subtype** set to EF_STREAM is used for experimental stream formats for development purposes only.

The 56 bits following the subtype are available to be used for any purpose.

Copyright © 2016 IEEE. All rights reserved.

Annex J

(normative)

IP encapsulation

J.1 Overview

This annex describes a method for encapsulating any Audio Video Transport Protocol Data Units (AVTPDU) in a User Datagram Protocol (UDP) datagram on an Internet Protocol (IP) network.

J.2 Encapsulation styles

There are two styles of UDP encapsulation defined:

a) Continuous

b) Discrete

The style of UDP encapsulation used for an AVTPDU is determined by the encapsulation style assigned to the **subtype** field of the encapsulated AVTPDU. The mapping of **subtype** field value to encapsulation style is defined in Table 6.

J.3 IP AVTPDU Format

J.3.1 Overview

This subclause documents the format of the Internet Protocol AVTPDU (IP AVTPDU) datagram. The IP AVTPDU contains the following fields:

— **encapsulation_sequence_num** (encapsulation sequence number): 1 quadlet

— **encapsulated_avtpdu**: variable length

Figure J.1 documents shows these fields with offset zero (0) shown as the first octet of the Ethernet frame. The size of the IPv4 or IPv6 header is variable depending on IP extension headers.

Copyright © 2016 IEEE. All rights reserved.

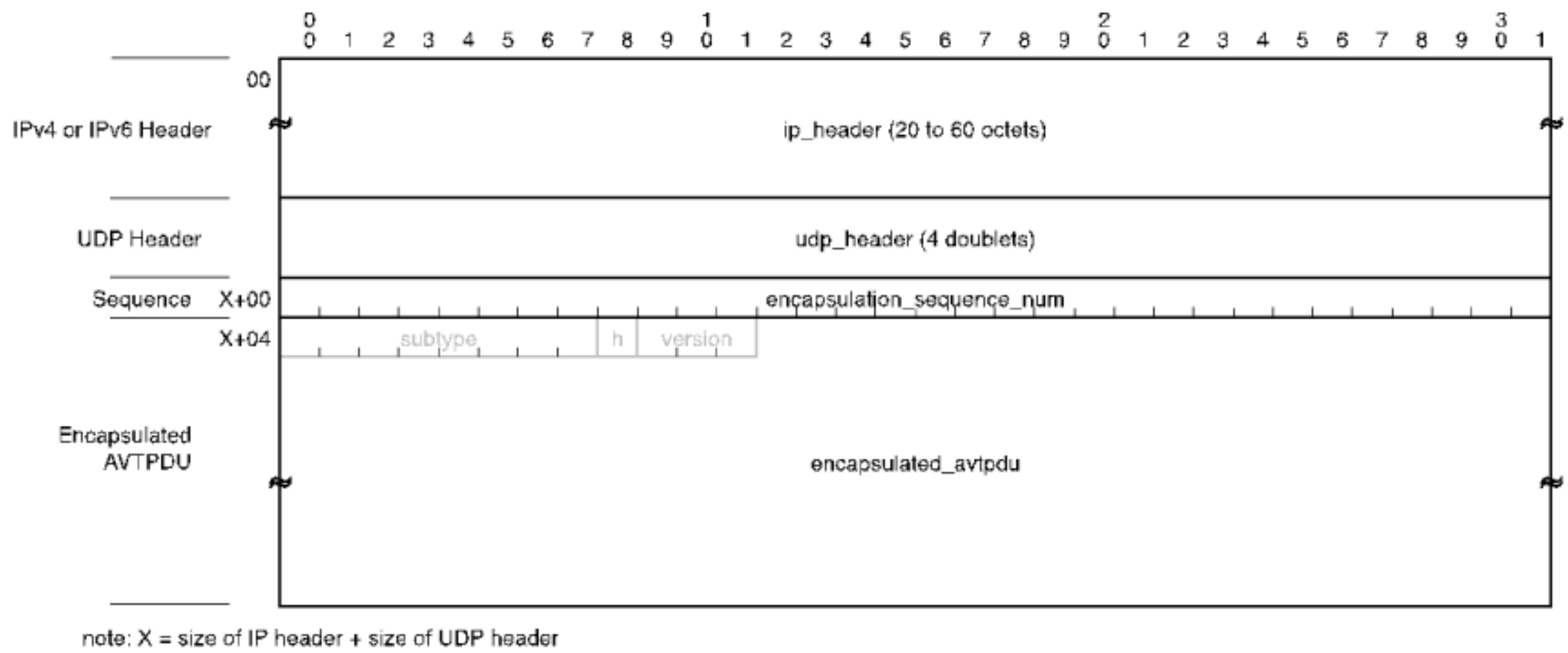


Figure J.1—IP AVTPDU datagram header fields

J.3.2 encapsulation_sequence_num field

The extended sequence number field (**encapsulation_sequence_num**) is a 4-octet field used to indicate the sequence of IP AVTPDUs sent from a source port of an end station to a specific destination IP address and IP port. This field may start at any value for the first IP AVTPDU sent and shall be incremented by one (1), with wrapping, on each subsequent IP AVTPDU with the same source IP address, destination IP address, source port, and destination port. This field is to wrap from $FFFFFFFF_{16}$ to 00000000_{16}.

The sender determines the initial value of this field. Listeners may use the extended sequence number to detect lost, duplicate, or out-of-order IP AVTPDUs.

An IP AVTPDU Listener shall be able to tolerate a starting **encapsulation_sequence_num** of any value.

NOTE—The encapsulation_sequence_num is useful for large networks with routers, where it is more likely that there will be more than 256 frames in transit between a Talker and Listener.

J.3.3 encapsulated_avtpdu field

The **encapsulated_avtpdu** field contains an AVTPDU as defined by 4.3.

J.4 Source port

The source port of a stream-related IP AVTPDU shall be an ephemeral port (as defined by RFC 6056) allocated by the application.

NOTE—The source port of the IP AVTPDU would therefore never be set to 17220 or 17221.

J.5 Destination port

The destination port of an IP AVTPDU depends on the encapsulation style used for the encapsulated AVTPDU.

Copyright © 2016 IEEE. All rights reserved.

— For the *continuous* encapsulation style, the destination port of the IP AVTPDU datagram shall be 17220 or an ephemeral port.

— For the *discrete* encapsulation style, the destination port of the IP AVTPDU datagram shall be 17221 or an ephemeral port.

J.6 Destination address

The destination address of an IP AVTPDU datagram may be a unicast IP address or may be a multicast IP address. When the destination address is a multicast IP address, the multicast address used shall be a transient application specific multicast address if a specific IANA standardized multicast address is not assigned for the value of the encapsulated AVTPDU's **subtype** field.

J.7 Bandwidth reservation

Stream Reservation Protocol (SRP) may be used to reserve bandwidth for streams of IP AVTPDUs. In order to use SRP for bandwidth management on an IP network, the IP AVTPDUs for the stream are tagged with the appropriate IEEE Std 802.1Q VLAN Tag field.

For each unique stream ID, the destination IP address of the streams IP AVTPDUs is a multicast IP address chosen such that the associated destination multicast MAC address does not conflict with any other multicast media access control (MAC) address used in the same Virtual Local Area Network (VLAN).

The mapping of multicast IPv4 addresses to multicast MAC addresses is defined in RFC 1112 Section 6.4.

The mapping of multicast IPv6 addresses to multicast MAC addresses is defined in RFC 2464 Section 7.

NOTE—The mechanisms used for bandwidth allocation, quality of service, and traffic shaping of IP AVTPDU streams on an IP network across LANs are beyond the scope of this annex.

J.8 Clock synchronization

All timing and synchronization requirements defined in 4.3 remain applicable when transmitting AVTPDUs in a User Datagram Protocol (UDP) datagram.

Copyright © 2016 IEEE. All rights reserved.

Annex K

(informative)

AES3 and AVTP Audio Format

This annex provides background information related to encapsulating AES3 streams via the Audio/Video Transport Protocol (AVTP) Audio Format (AAF). This includes AES3 streams containing non–PCM-formatted data in either Professional Format (SMPTE ST 337 and SMPTE ST 338) or Consumer Format (IEC 60958-3, S/PDIF).

NOTE—AES3 streams can be encapsulated in AVTP using either Clause 5 or Clause 7, however Clause 7 is preferred because it is explicitly designed for transporting both Professional and Consumer AES3 streams over AVTP, and does not involve the additional protocol requirements of Clause 5.

K.1 Definitions

K.1.1 AES3 Stream

AES3 Stream refers to the stream format defined by the AES3 set of standards.

K.1.2 AES3 Frame

AES3 Frame is as defined by the AES3 standard (see AES3-3-2009 Section 5). As shown in Figure K.1, AES3 streams are organized into blocks of 192 frames. Each frame consists of 2 subframes.

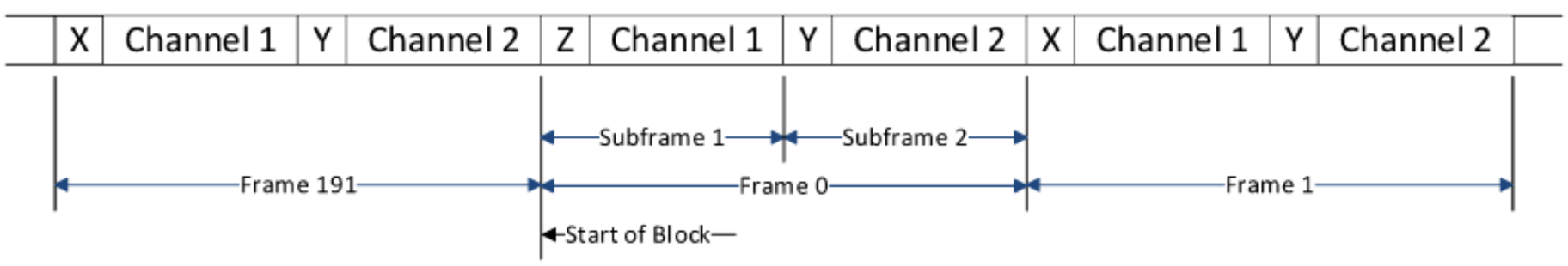


Figure K.1—AES3 Frame Format

The fields marked X, Y, and Z refer to AES3 Preamble fields whose functionality in AAF is replaced by block start markers (see K.2).

K.1.3 AES3 Subframe

AES3 Subframe is defined by AES3 (see AES3-3-2009 Section 4). As shown in Figure K.2, each 4-octet subframe consists of a single audio sample (nominally 24 bits) plus ancillary data (4 bits) plus protocol overhead (4 bits, marked Preamble in Figure K.2).

Copyright © 2016 IEEE. All rights reserved.

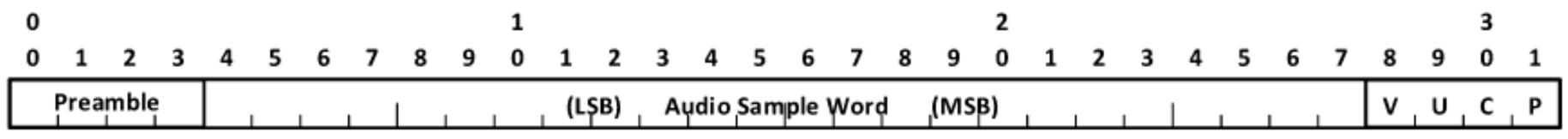


Figure K.2—AES3 Subframe Format

NOTE—AES3 subframes to be transmitted via AAF must be reformatted to match the AAF Subframe Format as described in 7.4.

K.1.4 AAF Subframe

AAF Subframe is defined by AAF, as shown in Figure 34 and described in 7.4. The fields of an AAF subframe are derived from, but are not identical to, the AES3 subframe (see Figure K.2).

K.1.5 Ancillary data

In AES3, ancillary Data refers to the bits V (validity), C (channel status), U (user), and P (parity). In AES47 (and AAF) the P bit is omitted and its location in the subframe replaced by the B (block start) bit (see 7.4.6.3).

K.1.6 Protocol overhead

Protocol overhead (as defined by AES47) refers to the data area marked as preamble in Figure K.2. The format of the preamble area is redefined by each of AES10 and AES47. In AAF the protocol overhead area is not used (compare Figure K.2 with Figure 34).

K.1.7 MADI

Multichannel Audio Digital Interface (MADI) is a serial interface (defined by AES10) that can carry 64 channels of audio (formatted as 32 AES3 streams) over a single link.

K.1.8 AVDECC Talker, Listener

AVDECC Talker is the part of a Talker that implements the AVDECC protocol. AVDECC Listener is the part of a Listener that implements the AVDECC protocol.

K.1.9 Block aligned

Block aligned describes two (or more) substreams that are being transmitted together and that have their block start markers in the same sample period. AES3 requires that the substreams within a single AES3 stream always be block aligned with each other. AES10 and AES47 do not require this, with the caveat that it makes the stream incompatible with AES3. For more info see K.2.

Copyright © 2016 IEEE. All rights reserved.

K.2 AES3 block start

K.2.1 Overview

AES3 requires that the frame that starts each 192-frame block of the AES3 stream (AES3-2-2009 Section 5.2) be marked.

AES10 and AES47 require a block-start marker on both subframe 1 and subframe 2, which allows greater independence of individual channels from each other. However, they refer to non-block aligned substreams within an AES3 pair as "most undesirable" because such a stream cannot be handled by an AES3 receiver. (See informative note at AES-10id-2005 (r2011) sec 6.2.4.)

Interfaces such as AES10 (and AAF) that transport more than one AES3 stream must allow that the AES3 streams are not block aligned with each other. For example, consider an interface device that has 32 AES3 inputs at the input side and 32 AES3 outputs at the other side, with MADI (or asynchronous transfer mode [ATM] or audio video bridging [AVB]) in between. Each incoming AES3 stream may come from a different source and thus cannot be expected to be block aligned with the other streams. (Note that alignment of media clocks between AES3 streams does not guarantee block alignment between those streams).

K.2.2 Channel status

The reason for the block start marking requirement is that each AES3 block conveys two 192-bit (24-octet) channel status data structures (CS) that are transmitted as a single data bit per subframe (the C bit). Thus knowing where the first bit of each CS begins is a requirement for reassembling the CS into its component fields and interpreting them correctly. The fields of the CS describe the content of the sample data in that substream (see AES3-2-2009 Section 5.5).

The U bit allows users to transmit the second (optional, user defined) 192-bit data block per frame.

The only purpose of the block start is to mark the beginning of each of the CS and user data structures.

K.2.3 AES3 block start implementation in AES3, AES10, and AES47

In AES3, the block start marking fields are based on the transmission format being a biphase serial link, which encodes the block start using specific 4-bit preamble values in the subframes (see AES3-3-2009, Section 4, Table 1).

In AES10 (also a serial protocol) the same area of the subframe containing the AES3 preamble is repurposed to provide the required block start information, but adapted to MADI (see AES10-2008, Section 4.2).

In AES47 and in AAF (both packet-based protocols) the block start marker is located in the ancillary data as the B (block start) bit, replacing the P (parity) bit (see 7.4.6.3 and K.6).

K.3 Bit ordering in AES3/AES10 compared with AES47/AAF

AES3 and MADI are hardware-based serial interfaces, and share similar strategy for block start and frame start. In addition, they transmit and receive data LSB first (see AES3-3-2009 Section 4.5). Most microprocessors and digital signal processors (DSPs) work on data structured with the MSbit at the MSbit

Copyright © 2016 IEEE. All rights reserved.

of the most-significant octet, which is located at the least-numbered address (or octet offset within a PDU) (see Figure 1 and AES47-2006-r2011, Section 4.1.2.1).

Thus data to be read directly from, or written directly to, a physical AES3 or AES10 interface must have the bit order of the data fields reversed to match AAF.

K.4 Audio sample, preamble, and ancillary data field order in AAF

AES3 and AES10 share the same order of these fields within their respective subframes, however AES47 uses a different order (see AES47-2006 [r2011], Section 4, 4.1, and 4.1.1).

The AAF subframe (see Figure 34) bears most similarity to (but is not identical to) AES47. The sequence of the fields in the AAF subframe is intended to make it simpler for microprocessor-based systems to manipulate the audio sample data field by positioning that field at the least-significant octets position of the 4-octet AAF subframe.

K.5 The Validity (V) bit

K.5.1 Overview

The AES3 Validity (V) bit is present in every subframe.

When the V bit set to zero (0) in a subframe, the format of that substream is PCM Audio (see AES3-1-2009 Section 3.4, Definitions, validity bit: "bit indicating whether the audio sample bits in the same subframe are suitable for direct conversion to an analog audio signal").

When the V bit is set to one (1), the format of that substream is not PCM audio, but some form of coded audio or other data.

See also AES-2id-2012 [B2], Section 4.7.

K.5.2 Professional and consumer formats

When the V bit indicates that the stream format is not PCM audio, then the Use of Channel Status Block field of the AES3 channel status structure (octet 0 bit 0) indicates whether the format of the non-audio data follows IEC 60958-3 (S/PDIF) (bit is zero [0], Consumer use) or SMPTE ST 337 (bit is one [1], Professional use). (See AES3-2-2009, Section 5.5.0, Channel Status Format, octet 0, basic audio parameters, note 1, and SMPTE ST 337 Section 5, Overview, Table 1).

There is some similarity between Professional and Consumer channel status formats, but they are not designed to be fully interoperable. The AAF **aes3_dt_ref** (aes3 data type reference) field provides a means to communicate this property in the AAF AVTPDU (see 7.4.5).

K.6 The Parity (P) bit

AES3-3-2009, Section 4.9, specifies that the P bit "shall carry an even parity bit such that time slots 4 to 31 inclusive will carry an even number of ones and an even number of zeros" ("time slot" in this context means "bit position in the AES3 subframe"). In AES3 and AES10 the P bit value is carried through from the incoming stream data, however in AES47 (and AAF) this bit is eliminated.

Copyright © 2016 IEEE. All rights reserved.

K.7 Bit ordering of ancillary data

The order of the ancillary data bits is also reversed (with respect to AES3/AES10) in both AES47 and AAF. Thus in AAF the ancillary data are sequenced B, C, U, and V.

K.8 SMPTE ST 337 data bursts

K.8.1 Overview

In SMPTE ST 337–formatted AES3 streams, the audio samples (data words) are formatted into data bursts each consisting of a burst preamble and a burst payload. The data burst words are the same size as PCM samples, and fit into the audio sample word field of the AES3 subframe, but the words contains a SMPTE ST 337 data stream, which has its own definitions of preambles, payload, channel status format, etc.

K.8.2 Data type values

The data_type field in the burst_preamble of each data burst defines the type of data carried within the burst_payload of the data burst. (See SMPTE ST 337:2015, Section 7.2.4.2). The data_type is a 5-bit field that can contain values as defined in SMPTE ST 338, for example the value indicating Dolby E is 28_{10}.

The AAF **aes3_data_type** field can be used to transmit a data type value in the AAF AVTPDU header, so that Listeners can determine the data type from the AAF stream without having to parse the payload stream (see 7.4.5). This field can also be used with S/PDIF data types (see 7.4.4).

K.8.3 Multiple data type values

AAF provides one **aes3_data_type** value per AAF AVTPDU header, however a single AAF stream can carry potentially hundreds of AES3 streams of varying data types. To communicate the data types of each of multiple AES3 streams in a single AAF stream to the Listener, AVDECC can be used as described in K.9. With AVDECC, the data type information is exchanged at the time the stream is configured, and is not transmitted with each AAF AVTPDU.

K.9 Using AVDECC to convey AES3 stream data types

A Listener can determine if a given audio channel in an AAF AVTPDU is an AES3 stream without parsing the payload because the **format** field of the AAF AVTPDU header will be set to **AES3_32BIT**. The Listener can then distinguish between (for example) “AES3 carrying PCM” and “AES3 carrying SMPTE ST 337–formatted Dolby E” by reading the AAF AVTPDU header **aes3_data_type** field (see 7.4.4) however use of this field is optional (see Table 16).

AVDECC enables an AVDECC Talker to define additional information for each of multiple AES3 streams in an AAF stream a stream, including (but not limited to) a data type value and a descriptive text string.

Figure K.3 shows an example stream routing within an AVDECC Talker, using AVDECC terminology.

Copyright © 2016 IEEE. All rights reserved.

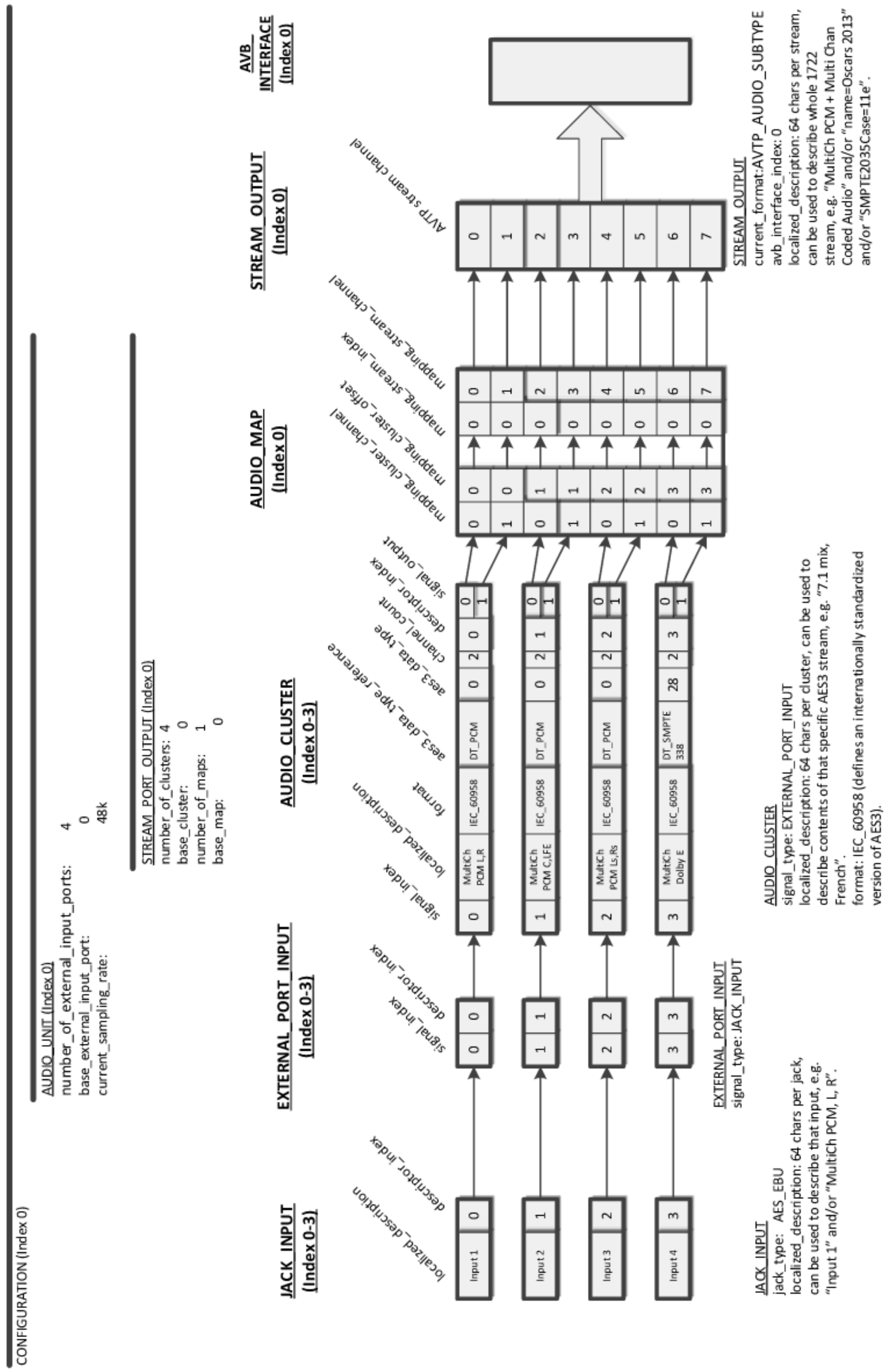


Figure K.3—AVDECC Entity Model example: four AES3 streams carried by one AVTP stream

Copyright © 2016 IEEE. All rights reserved.

The structure of Figure K.3 is based on the AVDECC Entity Model and its components (jacks, ports, clusters, audio maps, etc.) Each of these components and their interconnection can be defined by the AVDECC Talker using AVDECC descriptors such that they can be queried by AVDECC controllers using AVDECC messages.

Figure K.3 illustrates how per-stream data type values can be defined in the AVDECC AUDIO_CLUSTER through use of the AVDECC **aes3_data_type** and **aes3_data_type_reference** fields.

Figure K.3 also shows how an AVDECC Talker can define a name string (in UTF-8 format, up to 64 characters in length) for each individual AES3 stream that it can source. In addition the AVDECC Talker can define a name string describing the AAF stream as a whole (since it may consist of one or more individual AES3 streams). The content of these strings is defined between AVDECC Talker and AVDECC Listener (i.e., it is not specified by AVDECC). Examples of possible uses of the strings are included in Figure K.3.

Figure K.3 includes example values for relevant parameters of the AVDECC descriptors, illustrating their relationship to each other, and the specific parameter names. For further details on how to use these parameters see IEEE Std 1722.1.

AVDECC also provides other ways to convey information between AVDECC Talker and AVDECC Listener, for example by defining control types containing binary data fields (see 7.3.5.2.9 of IEEE Std 1722.1-2013).

K.10 Other considerations

K.10.1 AES3 multichannel mode

AES3-2-2009 Section 5.5.3 defines a way to associate a channel number with each AES3 substream in a group of AES3 streams. This mode is indicated within the ancillary data, and does not affect the setting of the AAF **channels_per_frame** field. If this feature is used, it may be desirable to also communicate this information to the Listener via AVDECC.

K.10.2 AES3 additional voice channels

AES3-2-2009 Annex B defines a way to carry additional channels of audio over a single AES3 stream (i.e., in addition to the left and right channels). This is done by using a subframe sample depth of 20 bits and using the remaining 4 least-significant bits of the 24-bit sample field to transmit a low bandwidth audio signal (i.e., 4 bits per sample). This mode is indicated within the ancillary data, and does not affect the setting of the AAF **channels_per_frame** field. If this feature is used, it may be desirable to also communicate this information to the Listener via AVDECC.

K.10.3 SMPTE ST 337 use of time multiplexing to transport multiple channels

SMPTE ST 337 provides a way to associate multiple substreams and transmit them in multiple bursts (see SMPTE ST 337:2008 Section 7.1.3.5). The bursts are associated together such that even though they arrive in the AES3 stream at different sample time offsets, the Listener plays the first sample of each burst (i.e., the first sample of each of the multiple channels) at the same time. (That time is defined through use of a timing or synchronization protocol.)

Use of this feature does not affect the setting of the **channels_per_frame** field or the bandwidth of the associated AES3 stream. In other words, multiple channels encoded into an AES3 stream using this scheme

Copyright © 2016 IEEE. All rights reserved.

use exactly the same total bandwidth as any other single AES3 stream. For example, four mono compressed audio channels could fit into the same (or fewer) number of samples (subframes) per second as two 48 kHz 24-bit PCM signals.

K.10.4 Varispeed

AES3 transport over AAF does not explicitly support AES10 Varispeed functionality (refer to AES-10id-r2011, Section 4, paragraphs 5 and 6).

K.10.5 Decoder latency

AVTP Presentation Time compensation for any latency incurred by decoding of SMPTE ST 337 data types must be managed by the application, and are not to be factored into the AVTP Presentation Time calculations (as described in 4.3.2).

K.10.6 AES3 high sample rate modes

AES3 defines some specific formatting modes for transporting PCM audio streams at higher sample rates, e.g., 88.2 kHz and 96 kHz (see AES3-2-2009, Section 5.5.1). This can result in a 96 kHz (mono) audio signal is being transmitted at a 48 kHz (stereo) data rate (e.g., by transmitting two samples each 48 kHz sample period).

In such cases the AAF **nsr** (nominal sample rate) field is set to match the AES3 data rate (i.e., not the internal stream PCM sample rate). The AES3 stream Channel Status structure communicates use of such modes to the Listener, but it may also be desirable to communicate this information via AVDECC).

K.10.7 SMPTE ST 337 Sample word length

SMPTE ST 337 supports audio sample (data) word lengths between 16 and 24 bits. For present day professional formats such as Dolby E, 20-bit is the norm, and some products have been released that support only the 20-bit and not the 24-bit word lengths. However, future formats are likely to take advantage of the extra bandwidth provided by the 24-bit word length, so implementers of new devices are advised to support the 24-bit word length.

K.11 Other references

AES-2id-2012 provides detailed information on use of channel status, and other important heuristics regarding implementation of AES3 and transport of AES3 through other transport systems (e.g., AES10).

Copyright © 2016 IEEE. All rights reserved.

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
© IEEE