IEEE P1722/Draft #0.08

Date: 2008-05-18

Draft Standard for Layer 2 Transport Protocol for Time Sensitive Applications in Bridged Local Area Networks

Sponsor:
Microprocessor Standards Committee (MSC) of the IEEE Society

Prepared by the Audio/Video Bridging Layer2 Transport Working Group of the IEEE Computer Society Microprocessors and Microcomputers (C/MSC)

Abstract:
<<Editor's note: TBD>>

Keywords:

<<Editor's note: TBD>>

Introduction

(This introduction is not part of IEEE P1722/D0.03 Draft Standard for Layer 2 Transport Protocol for Time Sensitive Applications in Bridged Local Area Networks.)

<<Editor's note: Additional introductory text TBD>>

Editors' Foreword

<<Notes>>

<<Throughout this document, all notes such as this one, presented between angle braces, are temporary notes inserted by the Editors for a variety of purposes; these notes and the Editors' Foreword will all be removed prior to publication and are not part of the normative text.>>

<<Comments and participation in IEEE standards development>>

<<All comments on this draft are welcome and encouraged. This includes not only technical comments, but also in the areas of IEEE standards presentation style, formatting, spelling, etc. as a properly formatted and structured document will improve the understanding and implementability of all relevant technical details. It is also requested that all technical and editorial comments should not simply state what is wrong, but also what in the commenter's opinion should be done to fix the problem.

Full participation in the development of this draft requires individual attendance at IEEE P1722 meetings. Information on P1722 activities, working papers, and email distribution lists etc. can be currently be found on the AVBTP Website: http://grouper.ieee.org/groups/1722/

Use of the email distribution list is not presently restricted to IEEE MSC members, and the working group has had a policy of considering ballot comments from all who are interested and willing to contribute to the development of the draft. Individuals not attending meetings have helped to identify sources of misunderstanding and ambiguity in past projects. Non-members are advised that the email lists exist primarily to allow the members of the working group to develop standards, and are not a general forum.

Comments and questions on this document may be sent to the AVBTP email exploder, to the editor or to the chair (email addresses below). Information on joining the AVBTP email exploder can currently be found at: http://grouper.ieee.org/groups/1722/reflector.html

PLEASE NOTE: Comments whose distribution is restricted in any way cannot be considered, and may not be acknowledged.

Editor of the P1722 Working Group:

Alan K. Bartky
SNAP networks
1871 SHARON LN
SANTA ANA CA 92705-5910
USA
+1 (714) 425 0967 (Tel)
EMAIL: alan@snap-networks.com

Chair of the P1722 Working Group:

Robert Boatright
Harman Pro
email: rboatright@ieee.org
+1 (801) 568-7566 office
+1 (801) 859-5294 mobile

>>

<<Editor's note: Document format and use of standard MSC templates for editing IEEE standards:

This document is being edited in Microsoft Word based on the 2007 standard MSC template and the goal of the working group is to adhere to the formats and conventions contained within the template and its associated guidelines document unless otherwise agreed to by the working group.

Standard MSC templates are available for Adobe Framemaker, Microsoft Word, and Open-Office. For more information and for additional web links to download the template files, please see:

http://grouper.ieee.org/groups/msc/WordProcessors.html

These templates are available under an open-source license; they may be used by any SDO, individual, or company. >>

Patents

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

For further information on IEEE patent policy, please see: http://standards.ieee.org/board/pat/pat-slideset.pdf

Participants

At the time this draft standard was completed, the Audio/Video Bridging Layer2 Transport Working Group had the following membership:

<< Editor's note: This list is based on active participants within recent history either by face to face meetings or by email contributions. At some point in the future, this will be replaced with an official list>>

Robert Boatright, Chair
Alan K. Bartky, Editor

Alan K. Bartky	**John Nels Fuller**	**Suman Sharma**
Alexei Beliaev	**Chuck Harrison**	**Kevin Stanton**
Robert Boatright	**Lee Minich**	**Michael Johas Teener**
George Claseman	**Matt Mora**	**Fred Tuck**
William Dai	**Dave Olsen**	**Niel Warren**
Craig Gunther	**Don Pannell**	**Andy Yanowitz**

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

(to be supplied by IEEE)

When the IEEE-SA Standards Board approved this standard on XX Month 200X, it had the following membership:

Name, Chair
name, Vice Chair
name, Secretary

(to be supplied by IEEE)

***Member Emeritus**

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

(To be supplied by IEEE)

Editor's name here
IEEE Standards Project Editor

Revision history

The following table shows the change history for this specification.

Version	Date	Author	Comments
0.01	2007-07-11	Alan K. Bartky	First version based on IEEE templates, IEEE other specifications and work to date from various contributions from http://www.avbtp.org
0.02	2007-07-13	Alan K. Bartky	Changed to MSC template. Misc. cleanup

0.03	2007-08-10	Alan K. Bartky	Summary changes:
			• Edited in initial proposals for Fragmentation/Reassembly, Cross-Timestamp functions.
			• Redesigned encapsulations to accommodate Fragmentation/Reassembly with standardized fields for fragment and packet lengths
			  o Added additional quadlet to all data stream packets
			  o Made it so all even numbered subtypes indicate "standard data stream header format, including 61883/IIDC and Proprietary/Experimental; and made odd numbered subtypes reserved for control
			    ▪ With that, changed reserved subtype value for AV/C control data from 2 to 3.
			    ▪ Also changed proprietary control messages to be subtype FF_{16} with subtype_data of zero (0).
			• Edited in changes based on comments from Cupertino face to face meeting.
			• Copied line numbering format from latest MSC template.
			• Added hyperlink, table of contents, references, etc. to output PDF file.

0.04	2007-08-27	Alan K. Bartky	• Edited in changes based on discussions at Santa Clara face to face meeting, 2007-08-23. • Added 64 bit Stream ID for all frames. Updated diagrams and text accordingly. • Added Control/Data bit after Ethertype as MSB and changed subtype field from 7 to 8 bits. Updated values, text and tables accordingly. • Edited in some of the changes for fragmentation and reassembly based on emails and teleconference discussions. Still more work to go here, but should be good enough to start discussing again. • Created new Annex Z for holding of issues and resolutions during creation of this specification. • Started editing some text from initial cut and paste bullets from my PowerPoint based contributions to be more "standard like" text. Still a lot more to go on this, but again hopefully still good enough to discuss at our meetings.

0.05	2007-11-29	Alan K. Bartky	• David V. James edited in suggested style, syntax, etc. type changes. Editor reviewed all of them and chose which to accept into the document (comments/question on those not accepted welcome). • Edited multiple encapsulation changes to realign the AVBTP frame to have the same quadlet alignment as IP packets (original alignment was based on trying to align the AVBTP packet in the same quadlet alignment as an Ethernet Frame, IP packets are actually not quadlet aligned with the frame as the Ethernet header is either 14 or 18 bytes long depending on if the frame is untagged (14) or tagged (18) format. Basically added two additional bytes at the start of each frame. • Changed control frame encapsulation to have a standard length and status field (i.e. took advantage of the 2 new bytes added). • Incorporated initial cut and paste and some modifications to John Nels Fuller's contribution on AV/C Command Transport Protocol. • Misc. cleanup, rewording and clarification of Fragmentation section (still more to work here). • Initial incorporation of Craig Gunther's contribution on AVBTP timestamp and 61883 SYT processing (mostly cut and paste, some minor edit's)
0.06	2008-03-07	Alan K. Bartky	• Edited in John's Fuller's contribution on Command Transport Protocol and did some misc. editing and cleanup. Also, per group consensus changed OPEN and CLOSE to OPEN and CLOSE • Edited in Dave Olsen's contributions on Timing/Synchronization and also MAC address allocation. For MAC address Allocation, created a new normative Annex (note: this may move to another IEEE 802 based standard in the future, but for now, we will keep working on this in P1722). o In process of editing the text in, did several changes from "should" to "shall" based on my understanding of the text and needs of the protocol as described by the text. The editor kindly requests others review all uses of "should" and "shall" throughout the document. • Updated AVBTP name list to be more reflective of those actually working and contributing to the standard. • Moved fragmentation/reassembly details from main text to Annex B as native AVBTP end stations will use CIP to do the breaking up of source data into AVBTP packets and will not support CIP packets that do not fit in a maximum size Ethernet frame. This work may still be useful for IEEE 1394 to AVBTP interworking units, so moving the text and diagrams there as a placeholder. • Added some placeholder text from some new assumptions as discussed at the Sandy Utah face to face meeting and entered into the draft assumptions document.

0.7	2008-04-07	Alan K. Bartky	• Updated editor's contact information • Updated Web page and email address info to official IEEE 1722 web site • Significantly updated Annex C based on discussions in IEEE 1722 and 802.1 AVB meetings. o Changed from "Multicast MAC address acquisition protocol" to "MAC address acquisition protocol" to reflect request to allow protocol to be expanded to also allow allocation of Source MAC addresses and to allow for a future DHCP like server for allocating MAC addresses. ▪ Based on that added new proposed fields and protocol operations o Added new tables and parameters to make it easier to tune. • Per discussions from last face to face meeting and also on teleconferences, Changed fields: o gm_info to gm_discontinuity o gm_generation to stream_reserved2
0.8	2008-05-18	Alan K. Bartky	• Removed Cross timestamp and Fragmentation from Annex B based on agreement to have 1394 to Ethernet AVBTP gateways use the source cycle time in the gateway_info field and require that 1394 to AVBTP gateways break up and reassemble as necessary on data block boundaries (i.e. Break up large CIP packets into multiple smaller CIP packets that fit on Ethernet and then reassemble as necessary when sending back on 1394 networks). • Changed diagrams and data definition to look more like IETF documents (bits 00-31, big endian format). Came up with a hybrid format that hopefully should please those familiar with 1394 conventions and/or IETF conventions. • Changed format and some supporting text on MAAP protocol in Annex C to: o Use start and length instead of start and end o Added renewal time field • Started some work on 61883 based timing and synchronization sections by going through the 61883 series of documents and putting in this document key sections that need to change to work in an AVBTP environment based on design needs and discussions to date.
			•

Table of contents

List of figures

List of tables

Draft Standard for Layer 2 Transport Protocol for Time Sensitive Applications in Bridged Local Area Networks

1. Overview

Increasingly, entertainment media is digitally transported. Streaming audio/video and interactive applications over bridged 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 A/V applications, etc. Technologies such as IEEE 1394, 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 streams, we lessen the effort of producing interworking units between IEEE 802 and other digital networks.

<<Editor’s note: The above text was copied from the IEEE P1722 PAR. This will be expanded or edited as appropriate.>>

1.1 Scope

This standard specifies the protocol, data encapsulations, connection management and presentation time procedures used to ensure interoperability between audio and video based end stations that use standard networking services provided by all IEEE 802 networks meeting QoS requirements for time-sensitive applications by leveraging concepts of IEC 61883-1 through IEC 61883-7.

<<Editor’s note: The above text was copied from the IEEE P1722 PAR. This will be expanded or edited as appropriate.>>

1.2 Purpose

This standard will facilitate interoperability between stations that stream time-sensitive audio and/or video across LANs providing time synchronization and latency/bandwidth services by defining the packet format and stream setup, control, and teardown protocols.

<<Editor’s note: The above text was copied from the IEEE P1722 PAR. This will be expanded or edited as appropriate.>>

1.3 Clauses

Clauses in this document are as follows:

Clause 1. Overview.

Clause 2. References

Clause 3. Terms, definitions, and notation

Clause 4. Abbreviations and acronyms

Clause 5. AVBTP base protocol

Clause 6. 61883/IIDC over AVBTP protocol

Clause7. Proprietary/Experimental subtype AVBTP protocol

Annex A. (informative) Bibliography.

Annex B (normative) Interworking 61883 between AVBTP and IEEE 1394 networks

Annex C (normative) MAC address Acquisition protocol

Annex Z (informative) COMMENTARY

2. References

The following standards contain provisions that, through reference in this document, constitute provisions of this standard. All the standards listed are normative references. Informative references are given in Annex A. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

[R1] IEEE Std 802®, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture [1]

[R2] IEC 61784-2:2007, Digital data communications for measurement and control – Part 2: Additional profiles for ISO/IEC 8802-3 based communication networks in real-time applications[2]

[R3] IEEE 802.3-2005, IEEE Standards for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and Physical Layer specifications

[R4] IEEE Std 802.1Q-2005, IEEE Standard for Local and Metropolitan Area Networks---Virtual Bridged Local Area Networks;

[R5] IEEE P802.1AS, IEEE standard for Local and Metropolitan Area Networks: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks;
<< draft document, see: http://www.ieee802.org/1/pages/802.1as.html>>

[R6] IEEE P802.1Qat, IEEE standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol;
<<draft document, see: http://www.ieee802.org/1/pages/802.1at.html>>

[R7] IEEE P802.1Qav, IEEE standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks - Amendment 11: Forwarding and Queuing for Time-Sensitive Streams;
<<draft document, see: http://www.ieee802.org/1/pages/802.1av.html>>

[R8] IEC 61883-1 (2003-01) Consumer audio/video equipment - Digital interface - Part 1: General;

[R9] IEC 61883-2 (2004-08) Consumer audio/video equipment - Digital interface - Part 2: SD-DVCR data transmission;

[R10] IEC 61883-4 (2004-08) Consumer audio/video equipment - Digital interface - Part 4: MPEG2-TS data transmission;

[R11] IEC 61883-6 (2005-10) Consumer audio/video equipment - Digital interface - Part 6: Audio and music data transmission protocol;

[1] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. New and revised IEEE standards and drafts are available for sale individually (http://shop.ieee.org/), and are also available, via an online subscription (http://standards.ieee.org/catalog/olis/index.html). The Get IEEE 802™® program (http://standards.ieee.org/getieee802/portfolio.html) grants public access to view and download current individual electronic (PDF) IEEE Local and Metropolitan Area Network (IEEE 802®) standards at no charge twelve months after publication.

[2] IEC publications are available from IEC Sales Department, Case Postale 131, 3, rue de Varembé, CH-1211, benève 20, Switzerland/Suisse. 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 . IEC publications are available for sale individually, and are also available via an online subscription (http://webstore.iec.ch/).

[R12] IEC 61883-7 (2003-01) Consumer audio/video equipment – Digital interface - Part 7: Transmission of ITU-R BO.1294 System B

[R13] IEC 61883-8 (work in progress) Consumer audio/video equipment - Digital interface - Part 8: Transmission of ITU-R Bt.601 style Digital Video Data

[R14] 1394 Trade Association TA Document 2003017 IIDC 1394-based Digital Camera Specification Ver.1.31[3]

[R15] IETF RFC 791 Internet Protocol (http://www.ietf.org/rfc/rfc0791.txt?number=0791)

[R16] <<Editor's note: TBD: Placeholder for other 1394TA documents>>

[R17] <<Editor's note: TBD: Placeholder for any IETF RFCs (if we need to refer to any in future versions of this spec)[4]>>

All the standards listed are normative references. Informative references are given in Annex A. At the time of publication, the editions indicated were valid.

<< Editor's note: Per MSC standards guidelines, need footnotes above on "how this document can be obtained and/or purchased", some of this work is TBD (will need info on how to obtain, IEC, 1394TA and possibly other documents). Also hopefully the boilerplate text from the template (2005) is still correct/OK>>

<<Editor's note: For some reason, my PDF converter tool is not creating hyperlinks in the PDF file if the hyperlink is in a footnote. For now, here are the links in an editor's note to make it easier for the reader to get to those web pages:>>

New and revised IEEE standards and drafts are available for sale individually (http://shop.ieee.org/), and are also available, via an online subscription (http://standards.ieee.org/catalog/olis/index.html). The Get IEEE 802™® program (http://standards.ieee.org/getieee802/portfolio.html) grants public access to view and download current individual electronic (PDF) IEEE Local and Metropolitan Area Network (IEEE 802®) standards at no charge twelve months after publication.

IEC publications are available for sale individually, and are also available via an online subscription (http://webstore.iec.ch/).

1394 Trade Association (1394TA) Members can download the 1394TA specifications for free from the members' website. Please note, however, that the copy right for each specification belongs to the 1394TA. Membership information can be found at: http://www.1394ta.org/About/Join/. For non-members, please contact jsnider@1394ta.org for information on how to obtain a copy of the 1394TA specifications and Technical Bulletins.

Internet Requests for Comments (RFCs) are available on the World Wide Web at the following URL: http://www.ietf.org/rfc.html.

[3] 1394 Trade Association (1394TA) Members can download the 1394TA specifications for free from the members' website. Please note, however, that the copy right for each specification belongs to the 1394TA. Membership information can be found at: http://www.1394ta.org/About/Join/. For non-members, please contact jsnider@1394ta.org for information on how to obtain a copy of the 1394TA specifications and Technical Bulletins . The mailing address for the association is at: 1394 Trade Association Office 1560 East Southlake Blvd., Suite 242, Southlake TX 76092 USA

[4] Internet Requests for Comments (RFCs) are available from the DDN Network Information Center, SRI International, Menlo Park, CA 94025 USA. They are also available on the World Wide Web at the following URL: http://www.ietf.org/rfc.html.

3. Terms, definitions, and notation

3.1 Conformance levels

Several keywords are used to differentiate between different levels of requirements and optionality, as follows:

3.1.1 expected: Describe the behavior of the hardware or software in the design models assumed by this specification. Other hardware and software design models may also be implemented.

3.1.2 may: Indicates a course of action permissible within the limits of the standard with no implied preference (“may” means “is permitted to”).

3.1.3 shall: Indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (“shall” means “is required to”).

3.1.4 should: An indication that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (“should” means “is recommended to”).

3.2 Glossary of terms

3.2.1 1 AVBTP communication: Information used in the operation of the AVBTP protocol, transmitted in an AVBTP message over an AVBTP communication path.

3.2.2 Audio/Video Bridging Transport Protocol. (AVBTP): 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.

3.2.3 AVBTP communication path: A segment of a network enabling direct communication between two or more AVBTP end stations.

3.2.4 AVBTP stream: An AVBTP stream is between one talker and one or more listeners

3.2.5 AVBTP port: A logical access point of an AVBTP clock for AVBTP communications to the communications network.

3.2.6 big endian: A method of transmitting a multi-byte integer. Bytes are transmitted in order of decreasing significance, i.e. the most significant byte is transmitted first.

3.2.7 byte: Eight bits of data, used as a synonym for octet.

3.2.8 controller: A device that introduces and manages talkers and listeners, and manages groups of sessions.

3.2.9 classA: P802.1Qav data stream traffic class with 125µs observation interval

3.2.10 classB: P802.1Qav data stream traffic class with TBD millisecond observation interval.

3.2.11 default: In this document the word default when applied to attribute values and options means the configuration of an AVBTP device as it is delivered from the manufacturer.

3.2.12 doublet: Two bytes of data.

3.2.13 epoch: The origin of a timescale.

3.2.14 event: An abstraction of the mechanism by which signals or conditions are generated and represented.

3.2.15 grandmaster selection time: The maximum amount of time required by 802.1AS to elect and propagate new grand master address.

3.2.16 holdover: A clock previously synchronized/syntonized to another clock (normally a primary reference or a master clock) but now free-running based on its own internal oscillator, whose frequency is being adjusted using data acquired while it had been synchronized/syntonized to the other clock, is said to be in holdover or in the holdover mode, as long as it is within its accuracy requirements.

3.2.17 holdover mode: When the 802.1AS clock is possibly instable due to a change in grandmaster a listener and talker shall revert to internal timing mode, ignoring the 802.1AS clock until the 802.1AS clock has once again stabilized.

3.2.18 ingress time: Ingress time is when the sample is sent by the talker application to the AVBTP layer. For example, on an I2S interface this is an 802.1AS timestamp of the word clock transition for the received sample.

3.2.19 link: A network segment between two IEEE 802 ports.

3.2.20 listener: A listener is a receiver of a stream.

3.2.21 maximum holdover time: The maximum time allowed for Grandmaster Selection plus clock stabilization on a listener.

3.2.22 multicast communication: A single AVBTP message sent from any AVBTP port and received and processed by all AVBTP ports on the same AVBTP communication path.

3.2.23 node: A device that can issue or receive AVBTP communications on a network**.**

3.2.24 octet: Eight bits of data, used as a synonym for byte.

3.2.25 octlet: Eight bytes of data.

3.2.26 port number: An index identifying a specific AVBTP port**.**

3.2.27 presentation time: Presentation time is the ingress time plus a delay constant

3.2.28 quadlet: Four bytes of data.

3.2.29 synchronized clocks: Two clocks are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of a single event at an arbitrary time differ by no more than that uncertainty. The timestamps generated by two synchronized clocks for the same event differ by no more than the specified uncertainty.

3.2.30 syntonized clocks: Two clocks are syntonized if they share the same definition of a second, which is the time as measured by each advances at the same rate. They may or may not share the same epoch.

3.2.31 talker: A talker is the source of a stream

3.2.32 timeout: A mechanism for terminating requested activity that, at least from the requester's perspective, does not complete within the time specified.

3.2.33 timescale: A linear measure of time from an epoch.

3.3 Unimplemented locations

The capabilities of all reserved, ignored, and unused values are carefully defined, to minimize conflicts between current implementations and future definitions.

3.3.1 reserved fields: A set of bits within a data structure that is defined in this specification as reserved, and is not otherwise used. Implementations of this specification shall zero these fields. Future revisions of this specification, however, may define their usage.

3.3.2 ignored location: Selected locations or portions of locations are partially implemented and are defined to be ignored. An ignored value has an affiliated storage element, but the value in the storage elements has no side effect.

3.3.3 reserved location: Some locations or portions of locations are not implemented and are defined to be reserved. When a reserved value is written, a zero values shall be assumed; when read, the returned value shall be ignored.

3.3.4 unused location: Selected locations or portions of locations may be not implemented or partially implemented and are defined to be unused. For unused locations, the selection between reserved and ignored behaviors is implementation dependent.

3.4 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 (0-9,A-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: MAC addresses and OUI/EUI values are represented as strings of 8-bit hexadecimal numbers separated by hyphens and without a subscript, as for example "01-80-C2-00-00-15" or "AA-55-11".

3.5 Notation of fields and values taken from other documents

<<Editor's note: For version 0.08, I have again reworked the notation section, this time to more follow conventions used in IETF RFCs as this document defines a transport layer and should not care about bit transmission order. So this document uses a merged convention to allow implementers who understand IETF RFCs and/or IEEE 1394 and/or 1394 Trade Association (1394TA) document conventions. This hopefully also should make it easier for implementers who wish to port their 1394/61883 technology to 802.1/802.3 Ethernet and 802.11 wireless networks or for implementers who have IP based technology to port/modify (i.e. if they can get the bit ordering correct for an IP packet, they can get it right for based on diagrams from this document). All comments on this change are welcome by the editor

It is also the editor's intent to take out unused template for areas not currently used by this draft (such as state machine conventions, descriptions on how to document registers, etc. If it turns out that we need to add any of these types of items in future versions of the spec, then these portions of the template can be put back into the document.>>

This document uses fields and values defined in other documents with multiple methods of defining such things as usage of upper and lower case, usage of underscore characters, italics, etc. As this document is intended to use these multiple protocols, its additional intent is also to make it easier for readers and implementers of those documents by not using different names and notation for those fields and values. So the following conventions are used for field names from other documents to match the convention from those documents.

a) Fields from IEEE 802.1Q: Fields are in all uppercase, no underscores (examples: DA, SA, TPID, CFI, VID)

b) Fields from IEEE 1394: Fields in lower case except for acronyms within the field name with optional underscores (examples: tcode, data_length, 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: DBC, DBS, Rsv).

3.2.2 Bit, byte, doublet, quadlet and octlet ordering

Similar to IETF RFC 0791 [R14] Internet Protocol (IP), this protocol is agnostic to the underlying bit order used by layers below it. Therefore all frame and packet formats contained within are specified as a series of 8 bit bytes where the actual transmission, reception, storage and retrieval of bits within the bytes are machine and/or lower layer specific.

This document uses the same convention as IEEE 1394 for abbreviating the following terms:

a) lsb: least significant bit

b) msb: most significant bit

c) LSB: least significant byte

d) MSB: most significant byte

Like Internet Protocol, the actual ordering of multiple bytes to store larger numbers or arrays of data is specified in big-endian order where the first byte of a multi-byte number is the most significant byte and the last byte is the least significant byte.

The significance of the interior bits within a byte uniformly decreases in progression from msb to lsb where msb is labeled as bit 0 and lsb is bit 1 and is shown in this document in a left to right order as follows.

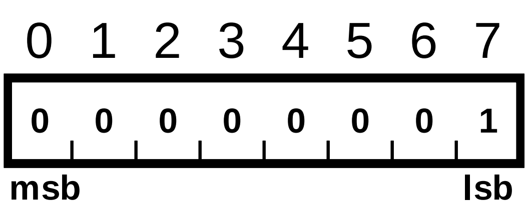


Figure 3.1 - Bit ordering within a byte

For the above figure, this would represent a decimal value of 1, a hexadecimal value of 01_{16} and a binary value of 00000001_2

This protocol specifies that all data to be transmitted and received for control and data frames shall always be using an integral number of 4 byte quadlets. A quadlet is a series of 4 bytes within a quadlet, the most significant byte is that which is transmitted first and the least significant byte is that which is transmitted last, as shown below.

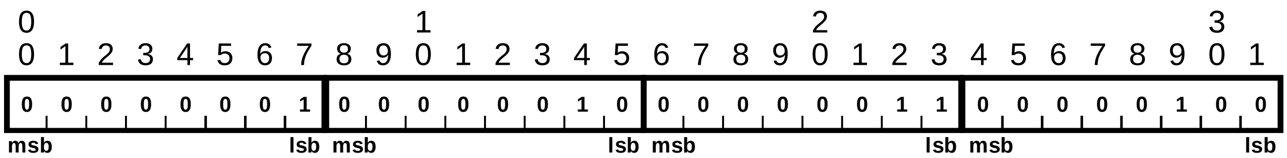


Figure 3.2 - Byte ordering within a quadlet

For the above figure, this would represent a quadlet holding four byte array of 1, 2, 3, 4 decimal.

A quadlet may contain bit fields of any length between 1 and 32 bits transmitted or received as a series of bytes. When a field spans more than one byte, the point where it spans the byte is shown as a large tick mark as follows:

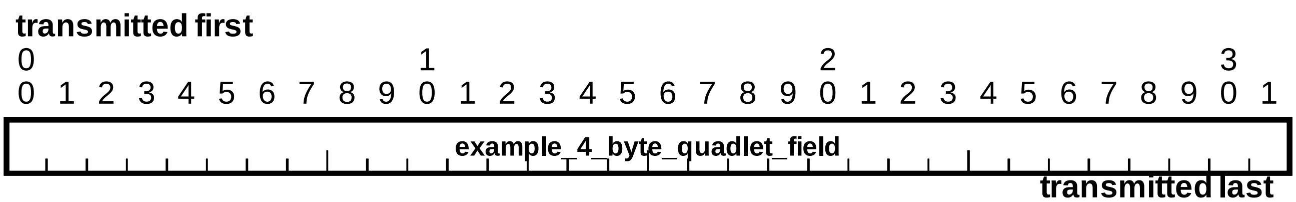


Figure 3.3 - Example 4 byte quadlet field diagram

For 64 bit fields that need to be contained in more than one quadlet, they are still transmitted and received as a series of 8 bytes, but for this document are shown in diagrams as follows:

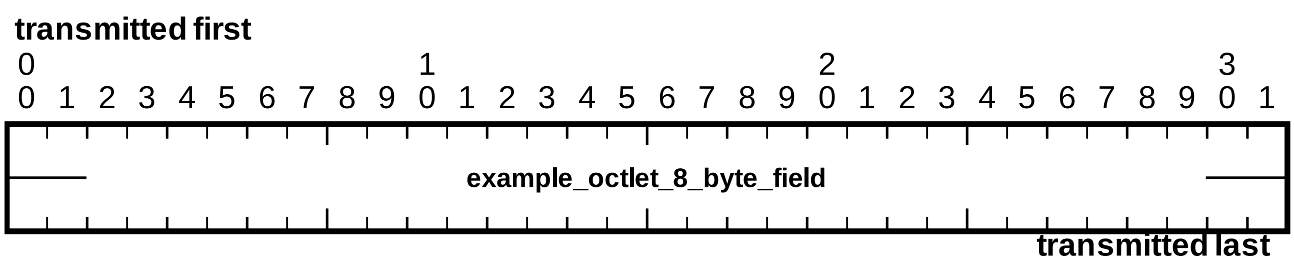


Figure 3.4 - Example octlet 8 byte field

When block transfers take place that are not quadlet aligned or not an integral number of quadlets, no assumptions can be made about the ordering (significance within a quadlet) of bytes at the unaligned beginning or fractional quadlet end of such a block transfer, unless an application has knowledge (outside of the scope of this specification) of the ordering conventions of the other bus.

3.5.1 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 3.1. A symbolic name, consisting of upper case letters with underscore separators, allows other portions of this document to reference the value by its symbolic name, rather than a numerical value.

Table 3.1wrap field values

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

Unless otherwise specified, reserved values are reserved for the purpose of allowing extended features to be defined in future revisions of this standard. Devices conforming to this version of this standard do not generate reserved values for fields, and process fields containing reserved values as though the field values were not supported. The intent is to ensure default behaviors for future-specified features.

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

3.6 Informative notes

Informative notes are used in this standard to provide guidance to implementers and also to supply useful background material. Such notes never contain normative information, and implementers are not required to adhere to any of their provisions. An example of such a note follows.

NOTE—This is an example of an informative note.

4. Abbreviations and acronyms

This document contains the following abbreviations and acronyms:

1394TA	IEEE 1394 Trade Association (www.1394ta.org)
IEEE	Institute of Electrical and Electronics Engineers, Inc. (www.ieee.org)
ACK	acknowledge
ANSI	American National Standards Institute (www.ansi.org)
AP	(wireless LAN) access point
AV	audio/video
AVB	audio/video bridging
AVBTP	audio/video bridging transport protocol
AV/C	audio video control protocol (from 1394 Trade Association)
BC	boundary clock
BMC	best master clock
BMCA	best master clock algorithm
BSS	basic service set
cd	control/data
CFI	canonical format indicator
CID	channel identifier
CIP	common isochronous packet
cntl	control
CoS	class of service
CRC	cyclic redundancy check
CTP	command transport protocol
D	draft
DA	destination MAC address
DRM	digital rights management
DSS	distribution system service
DTCP	digital transmission content protection (www.dtcp.org)
DTLA	Digital Transmission Licensing Administrator (www.dtcp.org)
DVCR	digital video-cassette recorder
E2E	end to end
EISS	enhanced internal sublayer service
ESS	extended service set
EUI	IEEE Extended Unique Identifier
fc	fragmentation control
GASP	global asynchronous stream packet

GM	grandmaster
GMT	Greenwich mean time
GPS	global positioning (satellite) system
HD	high definition
hdr	header
IEC	International Electrotechnical Commission (www.iec.ch)
IEEE	Institute of Electrical and Electronics Engineers (www.ieee.org)
IETF	Internet Engineering Task Force (www.ietf.org)
IP	Internet protocol
IS	integration service
ISO	International Organization for Standardization (www.iso.org)
IWU	interworking unit
kHz	kilohertz (thousand cycles per second)
LAN	local area network
LLC	IEEE 802.2 logical link control
LLDP	IEEE link layer discovery protocol
LSB	least significant bit
LMI	layer management interface
M	mandatory
MAAP	MAC address acquisition protocol
MAC	media access control
MACsec	media access control security
MHz	megahertz (million cycles per second)
MPEG	Moving Pictures Expert Group (http://www.chiariglione.org/mpeg/)
MS	master to slave
MSB	most significant bit
MTU	maximum transmission unit size
N/A	not applicable
NTP	network time protocol (www.ietf.org/rfc/rfc1305.txt)
O	optional
OC	ordinary clock
OUI	IEEE organizationally unique identifier
P	preliminary
P2P	peer to peer
PAR	project authorization request
PCP	priority code point
PICS	protocol implementation conformance statement

PLL	phased lock loop
PTP	precision time protocol
QoS	quality of service
Rsv, res	reserved
S Bridge	IEEE 802.1AS bridge
SD	standard definition
SI	international system of units
SID	source identifier
SM	slave to master
src	source
SRP	stream reservation protocol
STA	(wireless LAN) station
TAI	temps atomique international (international atomic time)
TBD	to be done (or determined)
TC	transparent clock
TG	task group
TLV	type, length, value
TPID	tagged protocol identifier
TS	timestamp
tv	timestamp valid
UTC	coordinated universal time
VID	VLAN identifier
VLAN	Virtual Local Area Network
WG	working group
WLAN	wireless local area network
X	prohibited
XTS	cross timestamp

5. AVBTP base protocol

<<Editor's note: This section will define the "base protocol" such that 61883 type protocol, encapsulation, etc. will be an optional protocol to "run over AVBTP" and so that we can add additional new protocols in the future. This section is intended for formats, functions, etc. that are "common">>

5.1 Overview

<<Editor's note: Text TBD>>

5.1.1 General assumptions/operations

<<Editor's note: This section and subsections is still mostly cut and paste from the assumptions document and needs further refinement, will be edited in future drafts of this document>>

5.1.1.1 Link bandwidth utilization

AVB classA together with AVB classB traffic cannot use more than 75% of a link's bandwidth. The remaining 25% (or more) shall be reserved for non-AVB flows.

5.1.1.2 Functional device type names

AVBTP will have Talkers, Listeners and Controllers

- A Talker is the source of a stream
- A Listener is a receiver of a stream
- A Controller is a device that introduces and manages talkers and listeners, and manages groups of sessions.

Any physical device can be any combination of these

An AVBTP stream is between one talker and one or more listeners

5.1.1.3 Interoperation with 802.1 bridges

AVBTP will interoperate with AVB 802.1 bridges.

If a stream traverses a bridge that is not AVB 802.1 capable, than that stream's bandwidth cannot be guaranteed, so interoperation with non AVB capable bridges is beyond the scope of this standard.

5.1.1.4 Point to point operation

AVBTP will be able to run in a point to point fashion when two AVBTP end stations are connected directly via an IEEE 802.3 Ethernet connection.

<< Editor's question/comment: What about point to point wireless??>>

5.2 802.3 Media specific encapsulation

This section documents the specific generic encapsulation requirements when running AVBTP over IEEE 802.3 LANs. This covers the following fields:

a) Destination MAC address: 48 bits

b) Source MAC address: 48 bits

c) 802.1Q protocol header: 4 bytes consisting of:

1) **Tagged Protocol Identifier (TPID): 16 bits**

2) **Canonical Format Identifier (CFI): 1 bit**

3) **Priority Code Point (PCP): 3 bits**

4) **Virtual Local Area Network (VLAN) Identifier: 12 bits**

5) **AVBTP Ethertype: 16 bits**

For 802.1Q operation (VLAN tagged frames) the Ethertype field immediate following the source MAC address is known as the Tagged Protocol Identifier (TPID) field and is set to 8100_{16}. For this case the AVBTP Ethertype is at an offset 4 bytes past the start of this field.

Figure 5.1 shows an AVBTP frame encapsulated within an 802.3 frame with an 802.1Q header (also known as an 802.1Q VLAN Tag field):

<<Editor's note: The 802.3/802.1Q frame format diagrams in this section are drawn differently than the AVBTP frames in the document due to the fact that the Ethernet header is either 14 or 18 bytes long and therefore not aligned up on a quadlet boundary. It is also not the editor's intent to fully document these fields but instead guide the reader through the basics of what this fields are, what specific requirements of the field values and usage are to support AVBTP. All other format and usage of the fields that are 802.3 and 802.1Q specific will be referenced to rather than specified here. The diagrams themselves are therefore technically informational, as they provide some duplicate information provided in other standards. The editor welcomes comments on if he should move the diagrams to an information Annex or if it is OK to label the diagrams as informational, but the text as normative (i.e. the text will have to specify how to use the fields for AVBTP)>>

Figure 5.1 - AVBTP frame within an 802.3 frame with 802.11Q tag field

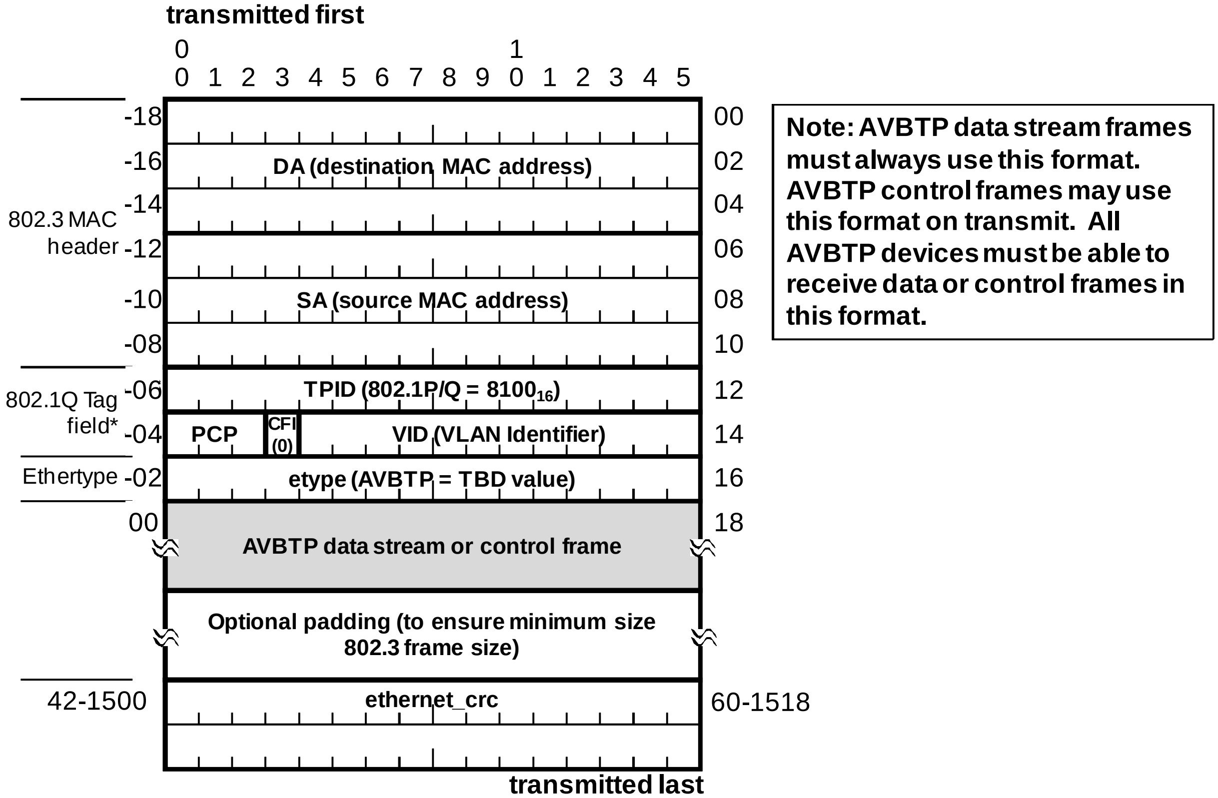
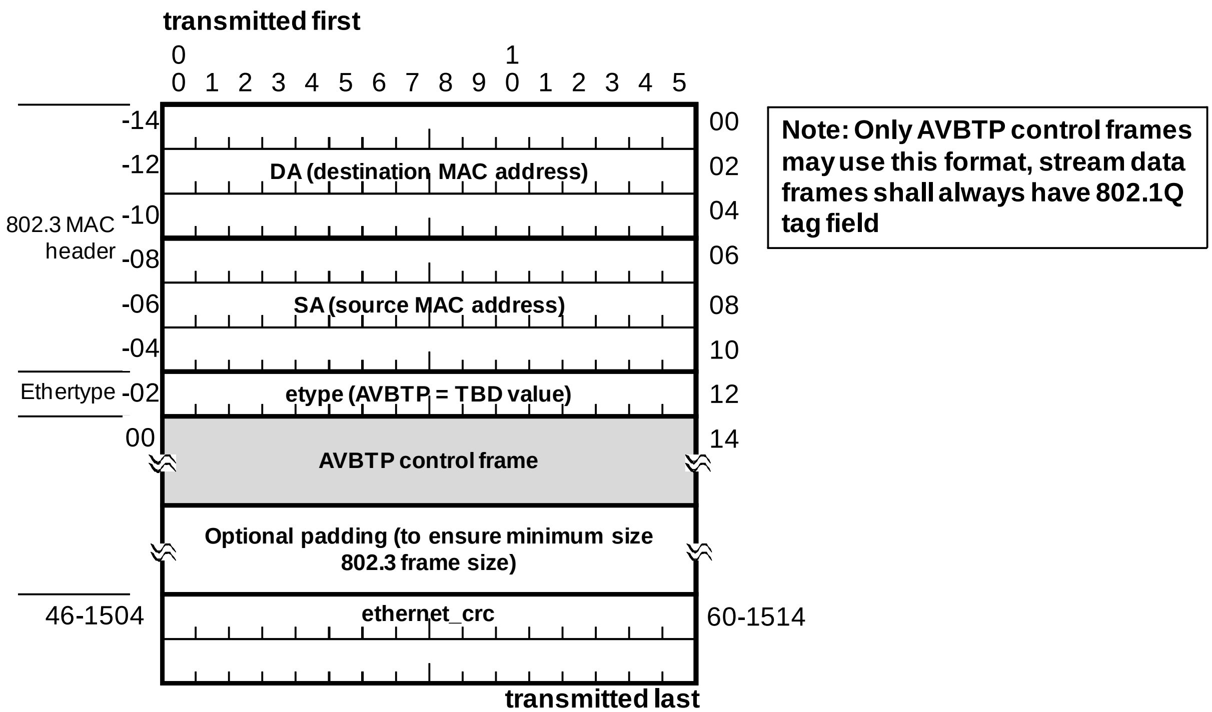


Figure 5.2 shows an AVBTP frame encapsulated within an 802.3 frame without an 802.1Q header:

Figure 5.2 - AVBTP frame within an 802.3 frame without an 802.11Q Tag field



5.2.1 802.3 Destination MAC address field

For AVBTP stream data frames, MAC Destination Addresses shall be unique for the Layer 2 network and may either be unicast or multicast addresses.

<< Editor's note: Need to modify/add text here now saying that for AVBTP stream data frames, we will require the use of 802.1Qat to allocate, free and generally manage both destination MAC addresses and stream IDs for individual AVBTP data streams.>>

For AVBTP stream control frames, MAC Destination Address may be unicast, multicast or broadcast depending on the specification of the usage of each AVBTP control frame.

5.2.2 802.3 Source MAC address field

For AVBTP stream data frames, MAC Source Addresses shall indicate the senders MAC address of the stream data or control traffic. Per IEEE 802.3 rules, this address shall always be a unicast MAC address.

5.2.3 802.1Q header field

Depending on the subtype of the AVBTP frame, the 802.1Q header may or not be required based on the following general rules:

a) All talkers shall send stream data frames (those frames with the cd bit set to 0) with an 802.1Q header present. This is due to the fact that the PCP field is required to indicate whether the stream is a ClassA or ClassB stream.

b) Talkers and controllers may send stream control frames (those frames) with an 802.1Q header.

c) All AVBTP compliant devices (talkers, listeners and controllers) shall be able to receive and process AVBTP data and control frames with an 802.1Q header present.

Additional rules for handling of 802.1Q headers may be listed in subsequent sections for current or future protocols that use AVBTP in current or future versions of this standard, but they shall not violate the above general rules.

The following rules shall apply for fields in the 802.1Q header if it is present:

5.2.3.1 802.1Q tagged protocol identifier (TPID) field

All frames with and 802.1Q header field shall set the TPID field (1^{st} Ethertype in the frame) 8100_{16} hexadecimal.

5.2.3.2 VLAN identifier

<<Editor's note: the text in this section is still in the format from my PowerPoint contribution, these needs to be put in IEEE standards style. Will do this in a future draft version of this document.>>

The VID is used to indicate a VLAN and is not to be used as a Stream Identifier

AVBTP stations shall be able to support a VLAN *Id* field value of zero to send or receive AVBTP frames.

AVBTP stations are recommended to support other VLAN *Ids*, but it is not required.

AVBTP stations not supporting VLANs must at least be able to process a received AVBTP frame with 802.1Q header and ignore the contents of that header.

If VLAN identification and knowledge is supported by an AVBTP station, it shall discard any received AVBTP frames with a VLAN ID for which it is not a member of the specified VLAN.

5.2.3.3 Canonical Format Indicator (CFI) field

For AVBTP, the CFI field shall be zero.

5.2.3.4 Priority Code Point (PCP) field

For data streams, AVBTP talkers shall set the PCP value to the 802.1Qat specified default values for stream classA traffic or TBD for classB traffic, unless they are changed from the defaults by a network administrator. As priority values may be changed by IEEE 802 bridges, the PCP value will be ignored on reception by AVBTP listeners.

<<Editor's note: Will probably insert some text here once a mechanism is specified on how to automatically determine what the PCP value and use it (at least refer to a future section that describes how to do this (probably will be via an 802.1Qat protocol mechanism (see Z.2 Need mechanism for getting PCP value for ClassA and ClassB streams)>>

<< Editor's proposal: AVBTP control traffic shall use the value as specified by the associated protocol specific value (e.g. 61883 over AVBTP), but never shall use a value of assigned for classA or classB traffic>>

5.2.4 AVBTP Ethertype, 16 bits:

<<AVBTP will have a single Ethertype value. That value will be put in this document per IEEE Ethertype assignment procedures, and not before then.>>

5.3 802.11 Media specific encapsulation

<<Editor's note: TBD>>

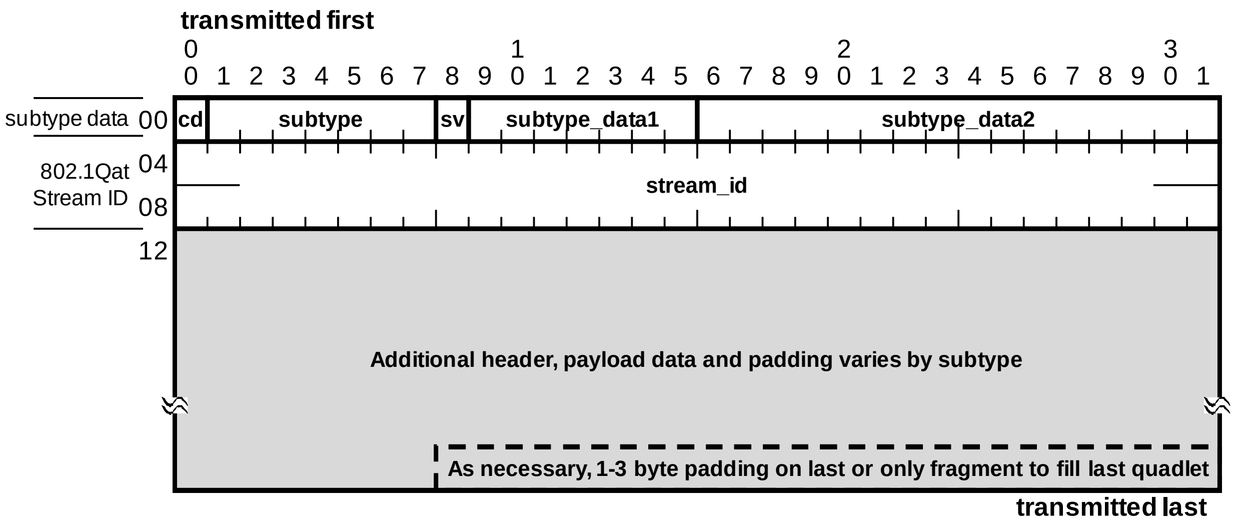
5.4 AVBTP frame common header format

This section documents the fields that are common to all AVBTP frames. This section documents the following fields:

a) First byte:
   1) **cd (control/data) field (cd) indicator: most significant 1 bit**
   2) **subtype field: remaining 7 bits**
b) Second byte:
   1) **sv (Stream ID valid) indicator: most significant 1 bit**
   2) **subtype_data1 field: remaining 7 bits**
c) subtype_data2 field: 16 bits
d) stream_id field: 64 bits

The following figure shows these fields encapsulated within an 802.3 frame with an 802.1Q header

Figure 5.3 —AVBTP frame common header fields



5.4.1 cd (control/data indicator) field

The cd bit indicates whether this AVBTP frame is a control or data frame

If the cd bit is zero, then this frame is an AVBTP stream data frame. See 5.6 below for additional encapsulation and protocol rules when this bit is set to zero. Only AVBTP talkers can set this field to zero as only talkers can send AVBTP stream data frames. If this field is set to zero, then for 802.3 frames, the talker shall ensure the frame is sent with an 802.1Q VLAN tag header present with the appropriate values for the TPID, PCP, CFI and VID fields.

If this field is set to one, then this frame is an AVBTP control frame. See 5.5 below for additional encapsulation and protocol rules when this bit is set to one. Any AVBTP station that sends control frames may set this bit to one.

5.4.2 subtype field

The 7-bit **subtype** field is used to identify the protocol running over AVBTP. Each protocol defines its use of AVBTP encapsulation within the rules established for common header formats for control and data frames.

Currently defined subtype values are listed in Table 5.1 below:

Table 5.1 -- AVBTP *subtype* values

Hexadecimal Value	FUNCTION	Meaning
00_{16}	61883_IIDC_SUBTYPE	61883/IIDC over AVBTP protocol
01_{16}-$7E_{16}$	-	Reserved for future protocols
$7F_{16}$	PROPRIETARY_SUBTYPE	Proprietary/Experimental over AVBTP

<<Editor's question: Should we go ahead and reserve a subtype for 1394 to AVBTP gateway control messages, or should we merge the XTS proposal and use one subtype for XTS and other 1394 gateway functions and/or needs>>

Subsequent parsing of AVBTP frames shall be based on a combination of the values contained within the subtype and cd fields.

5.4.3 sv field

The **sv** field is used to indicate whether the 64 bit stream_id field contains a valid IEEE 802.1Qat stream ID or not.

The bit is set to one if the stream ID is a valid stream ID

The bit is set to zero(0) if the stream ID is not valid.

For more details on valid combinations of the **stream_id** and **sv** fields see 5.4.6 below.

5.4.4 subtype_data1 field

The **subtype_data1** field consists of the remaining 7 bits of the byte containing the **sv** field and is used to carry protocol specific data based on the **subtype** and **cd** field values.

5.4.5 subtype_data2 field

The **subtype_data2** field consists of the two bytes (16 bits) following the subtype_data1 and is used to carry protocol specific data based on the **subtype** and **cd** field values.

5.4.6 stream_id field

If the sv field is set to one(1), then the stream_id field shall contain the 64 bit IEEE 802.1Qat stream ID associated with the frame.. This field shall be used for stream identification. The field shall be present in all AVBTP frames.

All AVBTP stream data frames shall contain a valid 64 bit IEEE 802.1Qat Stream ID in the stream_id field and shall set the sv (Stream ID Valid) bit to one(1).

AVBTP control frames relating to an individual stream shall contain a valid 64 bit IEEE 802.1Qat stream ID with the with the sv bit set to one.

AVBTP control frames not related to an individual stream should set the **stream_id** field to the NULL_STREAM_ID Value and shall set the **sv** bit to zero(0).

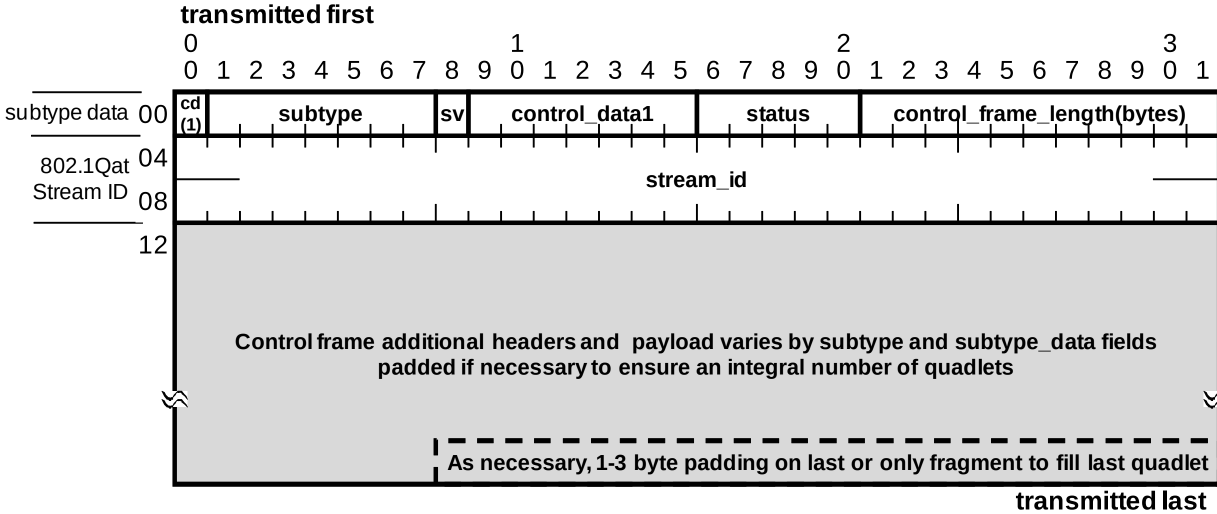
Note – Setting of the **stream_id** field to a consistent NULL_STREAM_ID is recommend instead of required as the NULL_STREAM_ID is intended for consistency and to avoid confusion for users such as those debugging AVBTP frame traces (i.e. so they don't see old data, or valid stream IDs, etc.). AVBTP end stations when receiving AVBTP frames with the **sv** bit set to one(1) shall ignore the entire contents of the **stream_id** field regardless of its value.

Valid stream IDs shall be allocated, managed and released using procedures as defined in IEEE 802.1Qat.

5.5 AVBTP common control frame header format

<<Editor's note: need introductory text for this section, here is the figure to start with>>

Figure 5.4 - Control frame common fields



5.5.1 status field

The 5 bit **status** field is available for use by the given control protocol as specified by the **subtype** field. If not used by the given control frame, then this field shall be set to zero (0).

5.5.2 control_data_length field

The 11 bit **control_data_length** field is used to contain the unsigned control frame payload length in bytes of all valid data bytes contained in the quadlets following the **stream_id** field in the AVBTP control frame header.

1 to 3 pad bytes shall be added at the end of the control frame payload area as necessary to ensure that an integral number of quadlets are in the control frame.

The maximum value for this field shall be 1488 decimal.

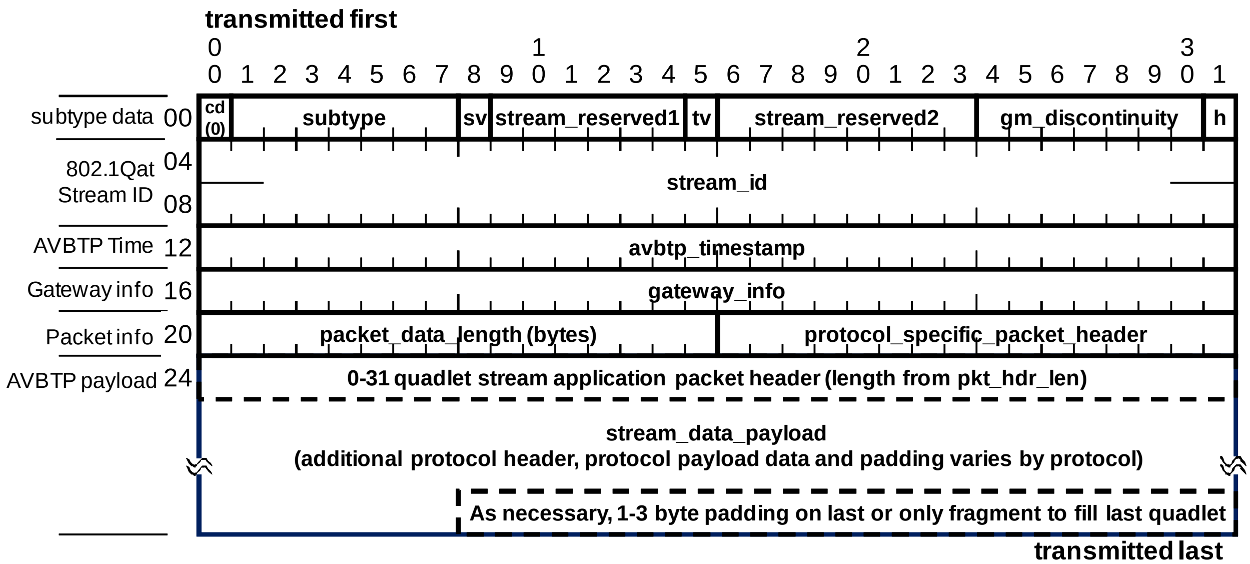
5.6 AVBTP common stream data frame header format.

AVBTP stream data is designed to standardize common use fields for source timestamping and for packet length. These fields are standardized for all AVBTP frames where the cd field is set to zero.

The AVBTP common stream data header format consists of the following fields after the subtype and in the following order:

a) **stream_flags** field: 8 bit byte with the following subfields defined when cd field equals zero:
   1) **sv (stream ID valid) most significant 1 bit of this byte**
   2) **tv (timestamp valid): least significant 1 bit of this byte**
   3) **stream_reserved1: remaining 6 bits of this byte**
b) **stream_reserved2**: 8 bits
c) **gm_discontinuity**: 7 bits
d) **h**: holdover: 1 bit
e) **stream_id** field: 64 bits
f) **avbtp_timestamp**: 32 bits
g) **gateway_info**: 32 bits
h) **packet_data_length**: 32 bits
i) **protocol_specific_packet_header**: 16 bits
j) **stream_payload_data**: 0 to n quadlets (where n does not exceed maximum frame size allowed by the layer 2 LAN)

Figure 5.5 --AVBTP common stream data header format (cd field set to zero)



5.6.1 subtype_data field subfields

The **subtype_data** field for AVBTP stream data (when the **cd** field is set to zero) has the following subfields defined.

- Most significant bit for **sv** (Stream ID valid) (see 5.4.3 above)
- Least significant bit for **tv** (timestamp valid) bit
- remaining 6 bits reserved (**stream_reserved1** subfield).

5.6.2 tv: (avbtp_timestamp valid) subfield

The source timestamp valid (**tv**) field is a one bit field used to indicate the validity of the **avbtp_timestamp** field time value.

If the timestamp valid bit is set to zero by the AVBTP talker, then this field shall indicate that the **avbtp_timestamp** field contains no data and therefore shall be ignored by an AVBTP listener.

If the timestamp valid bit is set to one by the AVBTP talker, then this field shall indicate that the **avbtp_timestamp** field is valid.

For how the **avbtp_timestamp** field is interpreted and processed see 5.6.6.

5.6.3 gm_discontinuity field

The **gm_discontinuity** field indicates a known or possible discontinuity in 802.1AS time. The **gm_discontinuity** field is stream specific. On stream creation the **gm_discontinuity** field shall be set to a random value. **gm_discontinuity** shall be incremented by 1 whenever a signaled or possible discontinuity is indicated from 802.1AS. These indications include, but are not limited to:

a) Discontinuity in absolute time
b) Discontinuity in frequency
c) Loss of Grandmaster clock
d) Election of new Grandmaster clock

5.6.4 h (holdover) field

Editor's note: this is a new concept and field as discussed at the Beaverton Oregon face to face meeting. This field needs better text and clarification to be added in the next version of the draft>>

The 1 bit **h** field shall be set anytime the AVBTP timestamp field in this stream is based off of some local clock instead of the 802.1AS based clock (i.e. in holdover clock based on the 802.1AS time from the previously known good grandmaster time). This is used by talkers to indicate that the timestamps may not globally synchronized with network time and is used by listeners to also use holdover time as necessary until all systems are probably running on the same grandmaster.

5.6.5 stream_id (802.1Qat stream identifier) field

The stream ID field is the same field as specified in 5.4.6 above. For AVBTP stream data frames, it shall always contain a valid 802.1Qat stream ID.

5.6.6 avbtp_timestamp field

The 32 bit **avbtp_timestamp** field shall express presentation time related to the 802.1AS Global Clock if the timestamp valid bit is set to one. The **avbtp_timestamp** represents the low order 802.1AS time converted to nanoseconds. The **avbtp_timestamp** rolls over approximately every 4 seconds.

If the source timestamp valid bit is zero, then the contents of the avbtp_timestamp field is undefined and should be ignored.

5.6.7 gateway_info field

This 32 bit field is used by gateway and interworking units to allow conversion and transport of audio/video data and control between AVBTP networks and other audio/video networks. One use is described in Annex B of this document for IEEE 1394 IEC 61883 to IEEE 1722 AVBTP IEC 61883 interworking.

Native AVBTP end stations not participating in this gateway function shall set this field to zero on transmit and ignore this field on receive.

5.6.8 packet_data_length field

The 16 bit **packet_data_length** field is to indicate the unsigned count of stream frame payload length in bytes of all valid data bytes contained in the quadlets following the **protocol_specific_packet_header** field in the AVBTP stream data frame header.

1 to 3 pad bytes shall be added at the end of the stream data frame payload area as necessary to ensure that an integral number of quadlets is in the control frame.

The maximum value for this field shall be 1476 decimal.

Note – This field is sized at a full 16 bits to allow for better interworking functions with protocols that support larger packet sizes such as IEEE 1394 but mandates a smaller maximum value to ensure that AVBTP frames can be transported across all IEEE 802 based networks.

<<Editor’s note: Need to add text recommend smoothing of AVBTP into similar size packets to make minimum packet size per interval better. This may be a good place to add text and/or we could have a separate section talking about packets per observation interval and how to best handle transmitting “evenly”.>>

5.7 Timing and synchronization

5.7.1 General

AVBTP defines a presentation time to achieve timing synchronization between talker and listener(s). The presentation time represents in nanoseconds the IEEE 802.1AS wall clock time when the data contained in the packet is to be presented to the AVBTP client at the listener(s).

AVBTP presentation time is used as a reference to synchronize any necessary media clocks and to determine when the first sample of a stream is presented to the client. Because media clocks vary with audio/video types the exact usage of the AVBTP presentation time is media format dependent.

5.7.2 AVBTP presentation time

The AVBTP presentation time is contained in the **avbtp_timestamp** field of AVBTP stream data frames.
The AVBTP presentation time may not be valid in every AVBTP packet. If an AVBTP packet contains a valid timestamp then the **tv** (Timestamp Valid) bit must be set to one.

The AVBTP presentation time represents the timestamp of the when the media sample was presented to AVBTP at the talker plus a delay constant to compensate for network latency. Unless otherwise negotiated between the talker and the listener the delay constant used to calculate the AVBTP presentation time is 2,000,000 nanoseconds (2 milliseconds).

The AVBTP presentation time as received by the listener(s) in **avbtp_timestamp** field should be utilized to synchronize the media clock of the listener to the talker. Since the AVBTP presentation time is directly related to the IEEE 802.1AS global time it may also be used to synchronize multiple talkers and listeners.

5.7.3 gm_discontinuity

Although the 802.1AS wall clock time is intended to be stable, it is possible for there to be discontinuities in the 802.1AS wall clock time. These could be due to events such as to changes the identity of the 802.1AS Grandmaster clock or changes in the timing source of the Grandmaster clock.

To aid in compensating for discontinuities in the 802.1AS time, all AVBTP stream data frames contain a **gm_discontinuity** field. The **gm_discontinuity** field is initialized to the random value on stream creation. The **gm_discontinuity** field of every subsequent AVBTP stream data frame shall contain the same value until a discontinuity is indicated by 802.1AS. When the actual or possible discontinuity occurs, the talker then increments the **gm_discontinuity** field by 1, after which all subsequent packets shall contain the new gm_discontinuity field. This process then repeats for every subsequent indication of an actual or possible discontinuity as indicated by IEEE 802.1AS.

When a talker detects a discontinuity, either from an 802.1AS indication or simple observation, it is required to increment the **gm_discontinuity** field by 1. This indicates to the listener(s) of the stream that the AVBTP presentation times contained in the **avbtp_timestamp** field may for a limited period of time not correspond to the 802.1AS wall clock and the listener should enter holdover mode.

When a listener detects that the **gm_discontinuity** field has changed or detects a discontinuity, either from an 802.1AS indication or simple observation, it should stop attempting to correlate AVBTP presentation time to 802.1AS wall clock time for one Maximum Holdover time. It is possible that the **gm_discontinuity** field could be incremented multiple times during one 802.1AS Grandmaster selection cycle. The listener should enter holdover mode, and begin timing the maximum holdover time, on the first indication of a discontinuity. If other indications of discontinuity are detected before maximum holdover time has expired, these indications shall be ignored until maximum holdover time has expired.

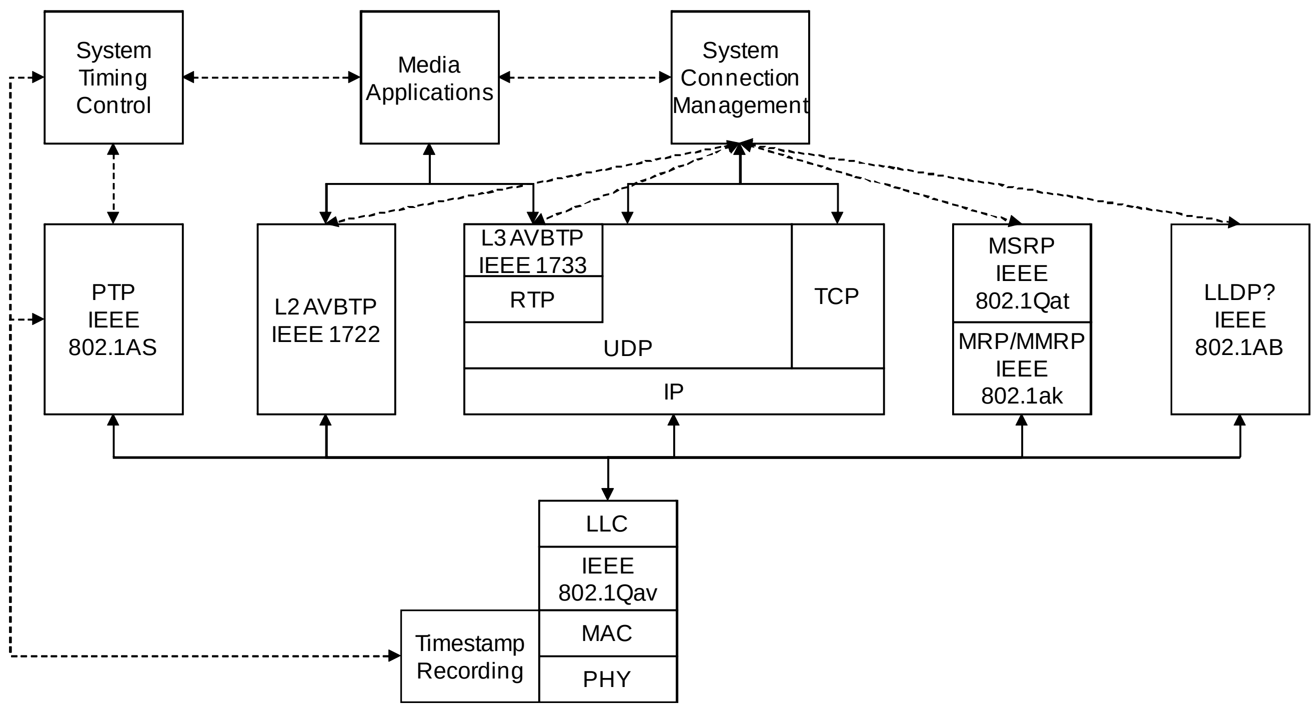
The listener should exit holdover mode after either maximum holdover time has expired or if the listener is able through observation to determine that the 802.1AS time and presentation time in the stream data are consistent with each other.

The value of the **gm_discontinuity** field is only meaningful to a single talker and its associated listener(s).

5.8 Protocol layering

<<Editors note: Text TBD. Purpose of this section is to document "common" layering for AVBTP and also describe to the reader how IEC-61883 and proprietary/experimental fits into the common layering model and how others can be added in the future. Below is the current working diagram I have for "common" layers. Also I have not labeled this diagram in that I'm not sure if it will stay, change, be removed, etc

<< Editor's note: It was agreed at the Sandy Utah meeting that 802.1BA will specify the layering and we will point to the BA spec. So the plan for documenting the lower layer is to change text in this section and remove sections defining lower layer protocols as BA will define this. For now, I am leaving this text in until BA becomes more mature.>>



5.8.1 Direct interfaces

The AVBTP common layer shall directly interface with the following protocols

- LLDP (802.1AB) **<<Editor's note: It may be that LLDP (802.1AB) may not be necessary for proper operation of AVBTP. If that is the case it will be removed. This diagram is planned to removed anyway once IEEE 802.1BA is more mature>>**
- SRP (802.1Qat)
- LLC (802.2), Ethertype option only (no length/DSAP/SSAP/etc. support).

5.8.2 Other layers needed to operate, but not directly interfaced with AVBTP

An AVBTP end station must also support the following protocols (that are in the system, but not directly interfaced with the AVBTP layer)

- PTP (802.1AS)
- 802.1Qav (queuing and scheduling)

5.9 Service interface

<<Editor's note: TBD>>

6. 61883/IIDC over AVBTP protocol

6.1 Overview

<< Editor’s note: text below for Overview for now just excerpts from our assumptions document. I have not changed everything to language “proper” for a standard. I should be able to do this in the next round of comments as this is the very first draft and I assume I’ll be doing a lot of major changes at least as far as moving things around). >>

<<AVBTP meeting note: To help simplify the work we are looking into making IIDC out of scope. Will post this to the reflector and solicit for comments. Editor will stop work on IIDC until he hears back from the team.>>

AVBTP adapts the following 1394/61883 type protocols to run in an IEEE 802 environment.

- 61883-2: SD-DVCR data transmission
- 61883-4: MPEG2-TS data transmission
- 61883-6: Audio and music data transmission protocol
- 61883-7: Transmission of ITU-R BO.1294 System B
- 61883-8: Transmission of ITU-R BT.601 style Digital Video Data
- IIDC

AVBTP replaces the following 1394/61883 type protocols with ones appropriate to an IEEE 802 environment.

- 61883-1: Function Control Protocol (FCP) is replaced with Command Transport Protocol (CTP)

6.2 Common 61883/IIDC Stream data encapsulation

The 61883/IIDC stream data encapsulation is used for carrying IEC 61883 and IIDC stream data traffic over AVBTP networks.

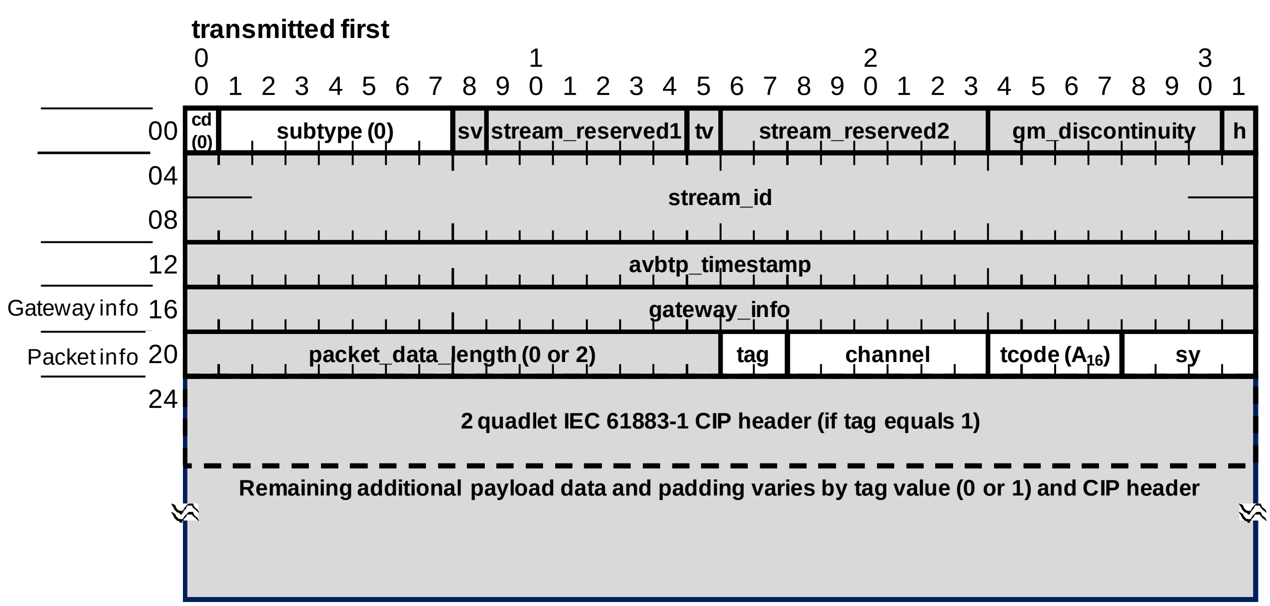
This encapsulation uses a **cd** field of zero(0) and a **subtype** field of zero(0).

This encapsulation also uses the **protocol_specific_packet_header** to contain 4 fields that are common for both IIDC and IEC-61883 frames. These fields are modeled after IEEE 1394 and consist of the following:

a) First byte:
   1) **tag field: most significant 2 bits of this byte**
   2) **channel field: least significant 6 bits of this byte**
b) Second byte:
   1) **tcode field: most significant 4 bits of this byte**
   2) **sy field: least significant 4 bits of this byte**

These fields are shown in the figure below:

Figure 6.1 61883/IIDC common header fields



6.2.1 tag field

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

Supported by AVBTP:

- 00_2: "data field unformatted" (used by Instrumentation & Industrial Digital Camera (IIDC) 1394 trade association specification)
- 01_2: CIP header is present

Not supported by AVBTP:

- 10_2: Reserved by IEEE 1394.1 clock adjustment
- 11_2: Global asynchronous stream packet (GASP) format (Used in 1394 for Serial Bus to Serial Bus bridges)

6.2.2 channel field

The 6-bit channel field follows the same meaning format and rules as specified by IEEE 1394. Of the four possible combinations for this field, the following values are supported as specified below.

- 0-30 & 32-63: originating channel ID from 1394 network via 1394/61883 to 1722/61883 gateway (as specified in Annex B below).
- 31: originating source is on AVB network (native AVB)

6.2.3 tcode (type code)

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

- AVBTP talkers shall always set this field to A_{16} hexadecimal on transmit
- AVBTP receivers shall always ignore this field on receive

6.2.4 sy field

Use of the 4-bit **sy** field is application specific and therefore beyond the scope of this standard. Known industry standards that currently use this field are:

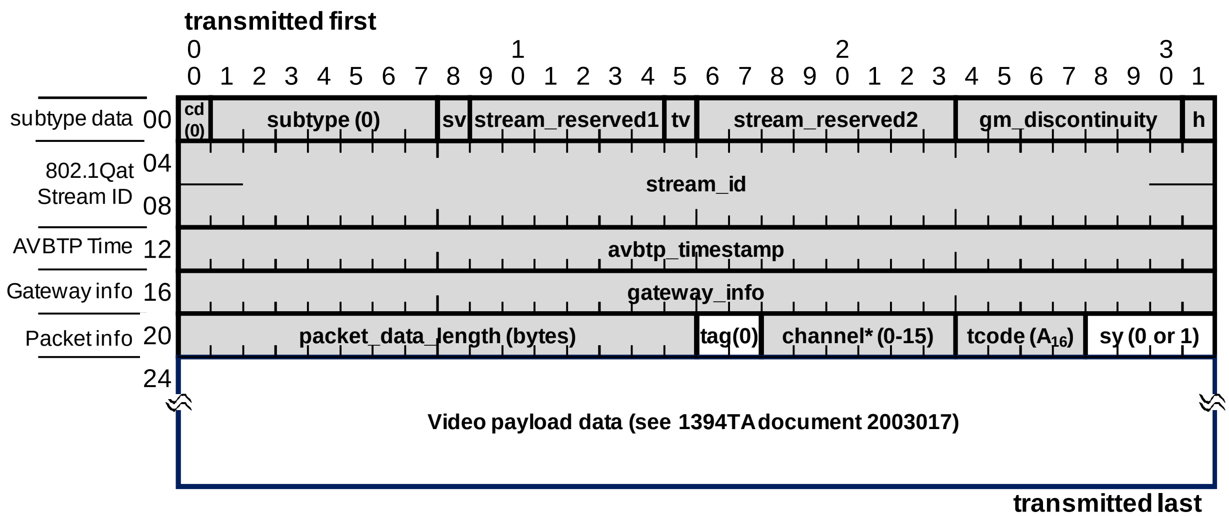
- IIDC [R13] (used for video start of frame indicator)
- Digital Transmission Content Protection (DTCP) (www.dtcp.com) [B3] [B4]

6.3 ”Data field unformatted” encapsulation (used by IIDC)

<< Editor’s note: This section and sub sections describe the variant where the tag field is set to 01_2 binary indicating that a CIP header follows. For tag value of 00_2 binary, the remaining data is format specific to IIDC (or other uses of “unformatted”??).>>

<< Editor’s note, this section is still cut and paste from my PowerPoint based contribution and will be re-worked to put in an appropriate standards language and format in a future version of this specification>>

Figure 6.2 - 61883/IIDC frame header fields



***Note: Current standard for IIDC restricts channel ID to 0-15**

6.4 IEC-61883 CIP encapsulation

<<Editor's note: Text TBD. Need to say something that the subsections of this section only apply if the tag field is equal to 01_2 binary. Also need general intro text>>

Figure 6.3—CIP header and data fields, tag= 1, SPH = 0

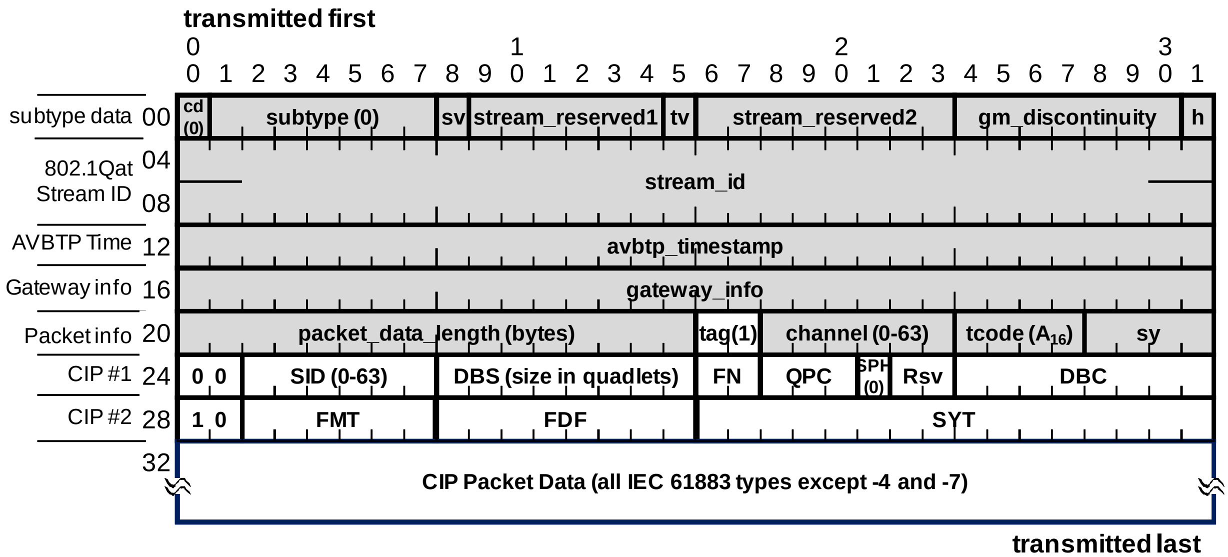
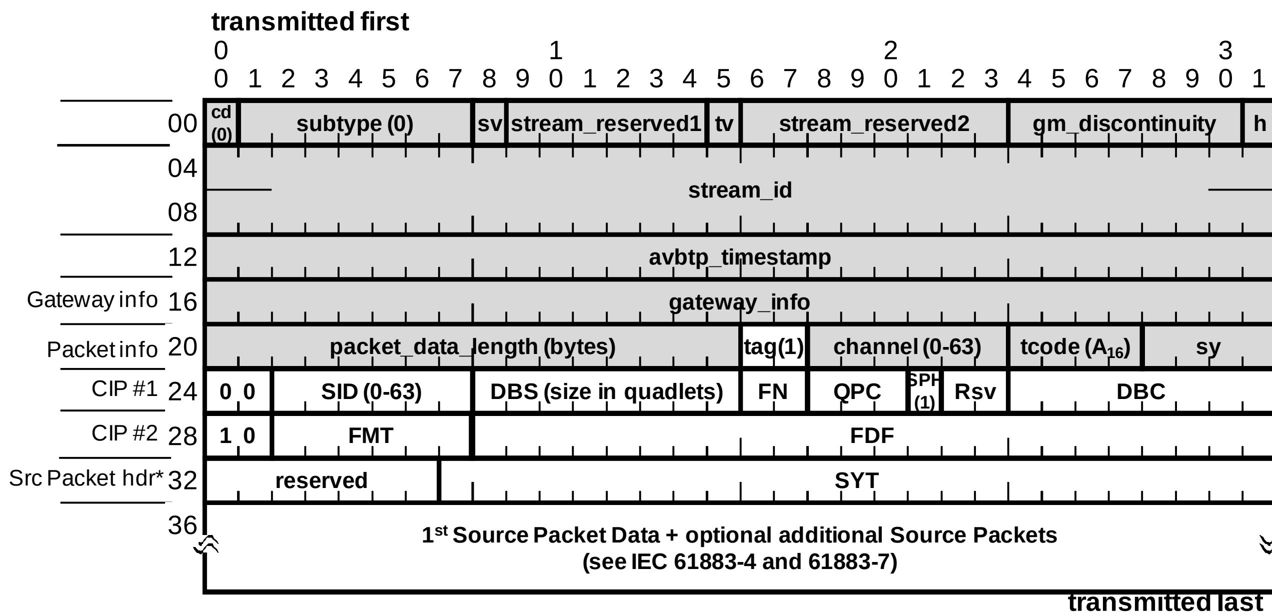


Figure 6.4—CIP header and data fields, tag=1, SPH = 1



6.4.1 CIP header 1st quadlet indicator

The 2 bit CIP header 1st quadlet indicator field has the same definition as defined in IEC 61883-1. AVBTP shall only support a fixed value of 00_2

<<Editors question: Is it OK for listeners to ignore this field on receive or should it be an error if there is another value??>>

6.4.2 SID (source ID) field

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

a) 0-62: originating Source ID from IEEE 1394 network (frame originated from a 1394/61883 to 1722/61883 interworking unit).

b) 63: originating source is on AVB network

6.4.3 Data Block Size (DBS)

The 8-bit DBS field has the same definition as currently in 61883-1, size of Data Blocks in Quadlets

- 0: 256 quadlets
- 1-255: 1-255 quadlets

6.4.4 QPC (quadlet Padding count)

The 3-bit **QPC** field has the same definition as currently defined in 61883-1. For all types of 61883 as defined today, this field is always zero.

<<Editor's note: Assume that we should say that this field is not supported in any current protocols as defined and therefore if QPC is not zero, it is an error>>

6.4.5 FN (fraction number) field

The 2-bit **FN** field has the same definition as currently defined I n 61883-1. This is currently only used in 61883-4 and 61883-7 (where also the SPH field is always set to one).

6.4.6 SPH (source packet header) field

The **SPH** bit has the same definition as currently defined in 61883. If set to one:

- Then application packet contains 61883-4 or 61883-7 (or future) source packets with source packet headers.

If set to zero

- Then application packet does not contain source packets (contains integer number of Data Blocks)

6.4.7 Rsv (reserved) field

The 2-bit has the same definition as currently defined in IEC 61883-1. It is reserved (currently not used by 1394/61883), set to zero on transmit, ignore on receive.

6.4.8 DBC (data block count) field

The 8 bit DBC field has the same definition as currently defined in IEC 61883-1. It contains the sequence number of the 1st data block in the packet. The DBC field shall not be set to a value that would cause the total byte AVBTP frame length to exceed a maximum payload size 802.3 of 1500 bytes (2 bytes of Ethertype plus 1498 bytes of payload) frame. This limits the total size of an AVBTP frame header plus payload of no more than 374 quadlets (1496 bytes). All frames of this format shall contain an integral number of data blocks as defined in IEC 61883-1.

6.4.9 CIP header 2nd quadlet indicator

The 2 bit CIP header 2nd quadlet indicator field has the same definition as defined in IEC 61883-1. For AVBTP it shall be fixed at 10_2 binary

6.4.10 FMT (stream format), field

The 6 bit **FMT** field has the same definition as currently defined in IEC 61883.

6.4.11 FDF (format dependent field)

The **FDF** field has the same definition as defined in IEC 61883. If the **SPH** field is set to 0, then this field is 8 bits in length. If the **SPH** field is set to 1, then this field is 24 bits in length.

6.4.12 SYT (synchronization timing) field (1394 cycle time based presentation time for SPH field equals 0)

The 24 bit **SYT** field is only present if the **SPH** field is set to a value of 1. This IEC 61883-1 defined field is present but not used by AVBTP end stations, it is only used by 1394/61883 to 1794/61883 interworking units (see Annex B below). AVBTP talker end stations shall set this field if present to $FFFFFF_{16}$ (the IEC 61883 no data timestamp value) on transmit and AVBTP listener end stations shall ignore this field on receive.

6.5 Command transport protocol (CTP) frame format

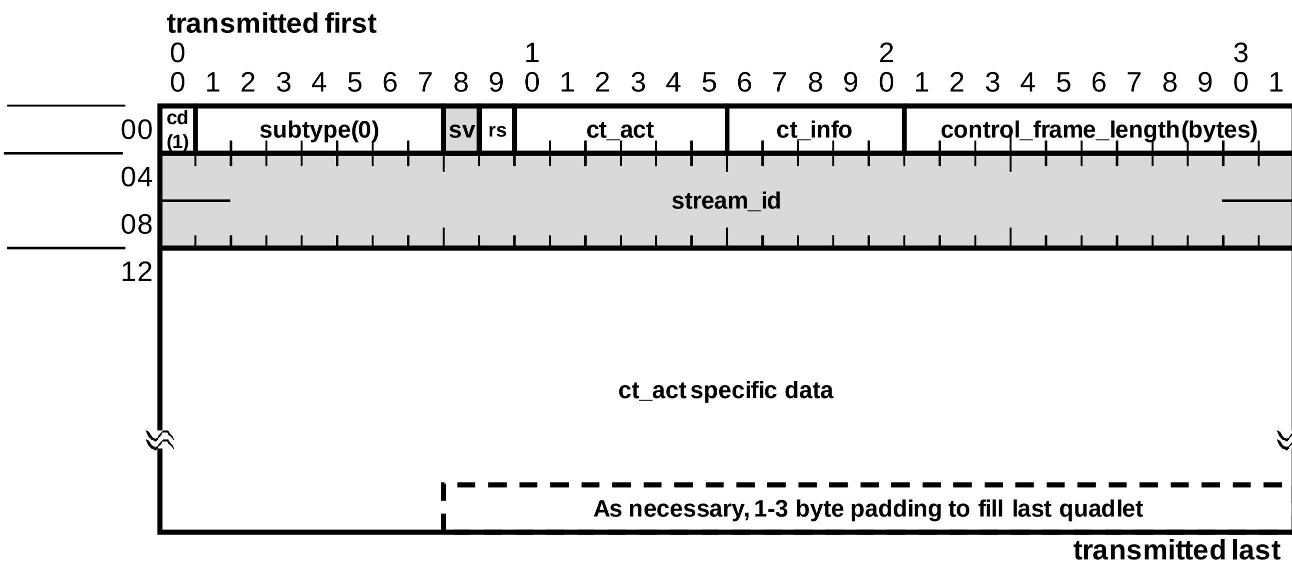
<<Editor’s note: This entire section may be removed in a future version of this specification as it is currently being discussed if we need this protocol, or if we will mandate that AVBTP end stations support IPv4 and/or IPv6 and use the AV/C over IP protocol as being defined by the 1394TA. It is the Editor’s opinion that at minimum we should at least not mandate use of CTP as implementers are already prototyping AV/C over IP wheras CTP has not been prototyped yet as of today. CTP was created to allow stations to not have to implement an IP protocol stack. Again, as IP is under discussion for other puposes, end stations may need to have an IP stack anyway. It is still group consensus to do everything we can for AVBTP to not require TCP protocol and have all IP based control protocols be IP based.

This section also in general needs more work, but the editor is holding off on any additional work until this work is agreed to be needed (i.e. that we will allow both AV/C over IP and AV/C over CTP)>>

This section defines the format and field values for subtype 0 control frames for use with 61883/IIDC data frames which is called the Command Transport Protocol.

An AVBTP control frame with a subtype of zero shall only be used for CTP. The common format of the frame is shown in the figure below:

Figure 6.5 - Command transport protocol common fields



cd field

The 1 bit **cd** field shall be set to 1 for all CTP frames

Subtype field

The 7 bit subtype field shall be set to 0 for all CTP frames

<u>**sv** field</u>

The 1 bit **sv** (Stream ID valid) field shall be set to 0 for all CTP frames

rs field

The **rs** field is a 1 bit field used to indicate if the frame was sent by a CTP requestor or a CTP target. A CTP requestor sending CTP frames, shall set this field to 0 (to indicate an *action*). A CTP target sending CTP frames, shall set this field to 1 (to indicate a *response*).

ct_act field

The **ct_act** field is a 7 bit field used to indicate the action to be performed by this frame. The currently defined valid values are defined in Table 6.1.

Table 6.1 – ct_act field values

Decimal Value	FUNCTION	Description
0	**OPEN**	Open connection to target
1	**CLOSE**	Close connection to target
2	**SEQ_RST**	Sequence number system reset
3	**EXECUTE**	Execute command set command
4	**EVENT**	Manage EVENT notification

ct_info field

The Command Transport Protocol information field (ct_info) is an 8 bit field used to provide one of the following types of information:

cmd_set_id:

<<Editor's note, need text here>>

cmd_status

<<Editor's note, need text here>>
The currently defined valid values are defined in Table 6.2.

reserved:

Field is reserved for future use. Set all bits in the field to zero on transmit. Ignore on receive.

Table 6.2 –cmd status field values

Decimal Value	FUNCTION	Summary Description (see the section for all CTP responses for any further interpretation of these status values)
0	**SUCCESS**	command executed successfully
1	**ALREADY_LOGGED_IN**	requesting station is already logged in, sequence numbers are reset
2	**NOT_LOGGED_IN**	no connection exists with requesting station, command ignored
3	**NO_RESOURCES**	not enough resources to establish a connection
4	**PENDING**	command accepted, but not completed
5	**NOT_SUPPORTED**	the specified cmd_set_id is not supported
6	**SEQ_NOT_EXPECTED**	the specified cmd_sequence was not in range
7	**EVENT_REGISTERED**	command executed successfully, an event has been registered, the event ID is in the extended_error field

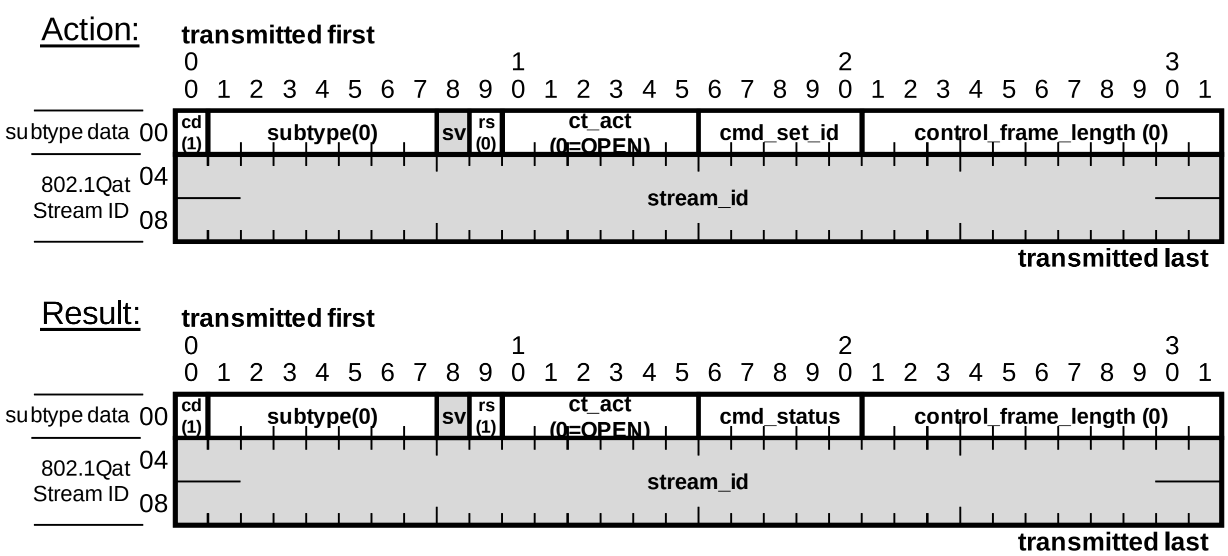
<<Editor's question: As CTP is porting across a protocol that currently runs under 1394 which all the protocols I've seen that use it do not do any 1 to 3 octet padding at the end. Are there any cases where AV/C will have a packet that needs to be padded to fill the last quadlet>>

stream_id

For all CTP frames, the **stream_id** is not used, and should be set to NULL_STREAM_ID on transmit.

6.5.1 OPEN action and result.

Figure 6.6 - OPEN command transport frames



Open action notes:

- **cmd_set_id** is CTP's status from the **<<Editor's note: text is missing here>>**
- **stream_id** is not used, set to NULL_STREAM_ID

Open result notes:

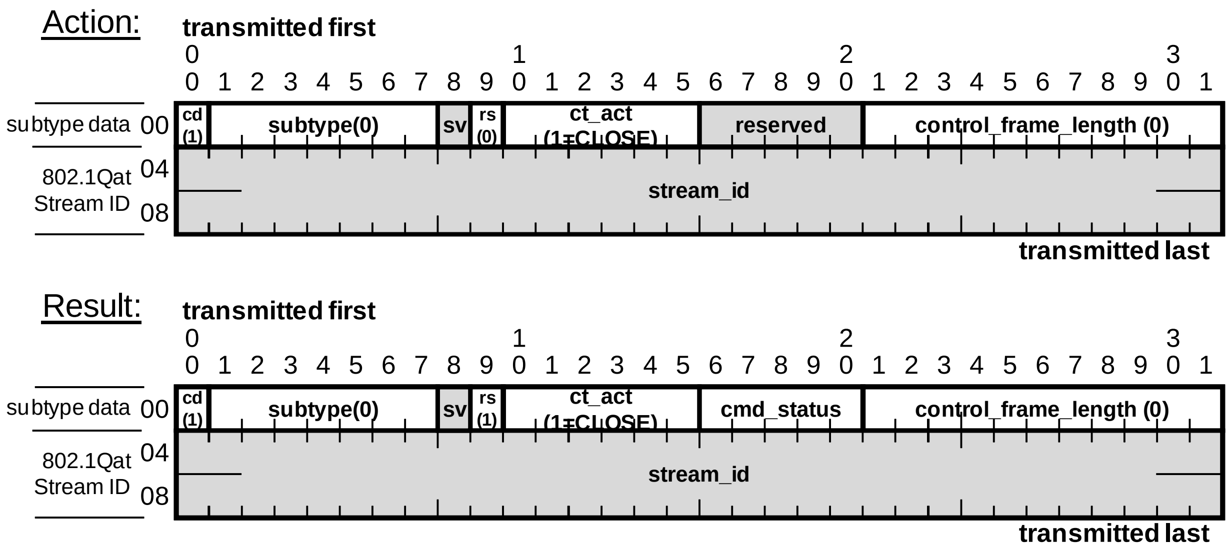
- **stream_id** is not used, set to NULL_STREAM_ID
- **cmd_status** is CTP's status from the action, values:
  - **SUCCESS**: no errors, sequence numbers are reset, target can store at least one EXECUTE result for this connection
  - **ALREADY_LOGGED_IN**: requesting station is already logged in, saved responses discarded, commands in process may or may not complete (but no response will be sent), **NextExpectedSequence** set to 1, **LastAcknowlegedSequence** set to 0, all event registrations are deleted.
  - **NO_RESOURCES**: not enough resources to establish a connection
  - **NOT_SUPPORTED**: target implementation does not support CTP

NOTE: If an initiator is already logged in then the OPEN action differs from the SEQ_RST action only in that it also deletes all event registrations at the target for the initiator.

<<Editor's note: should the Open result return information as to which command sets are supported??>>

6.5.2 CLOSE action and result

Figure 6.7 - CLOSE command transport frames



Close Action

Close Action Notes

- **stream_id** is not used, set to NULL_STREAM_ID

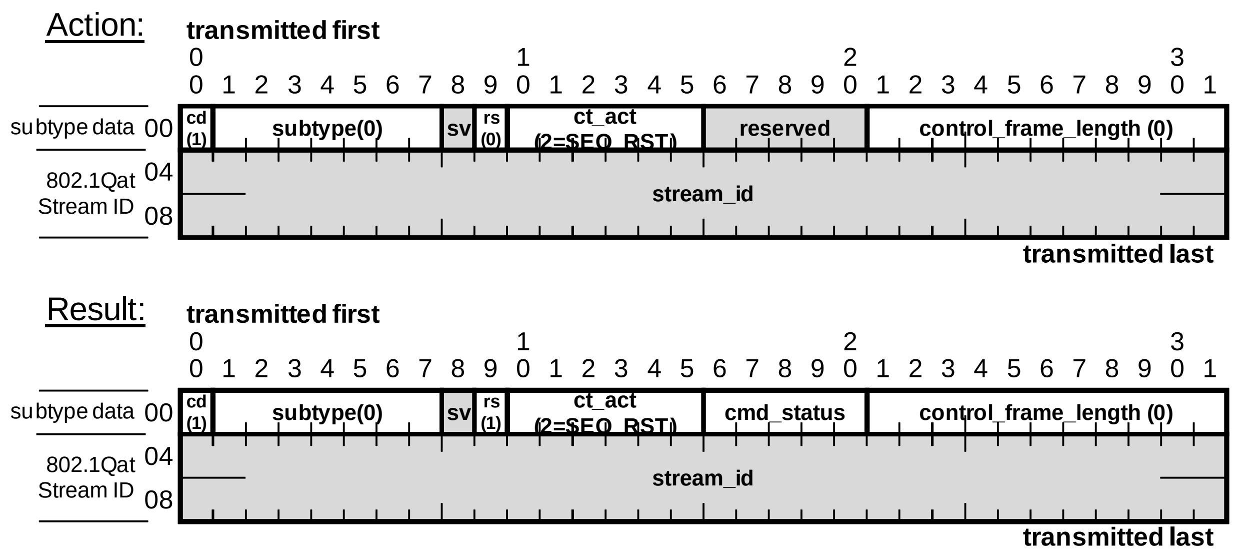
Close Result

Close Result Notes

- **stream_id** is not used set to NULL_STREAM_ID
- **cmd_status** is CTP's status from the action, values:
  - **SUCCESS**: no errors, connection terminated, commands in process may or may not complete (but no response will be sent), all event registrations are deleted
  - **NOT_LOGGED_IN**: no connection exists with requesting station

6.5.3 SEQ_RST (Sequence number reset) action and result

Figure 6.8 - Sequence number Reset command transport frames



SEQ_RST (Sequence number Reset) Action

SEQ_RST Action Notes

- Asks Target to discard any saved responses, set **NextExpectedSequence** to 1, set **LastAcknowlegedSequence** to 0
- **stream_id** is not used, set to NULL_STREAM_ID

SEQ_RST Result

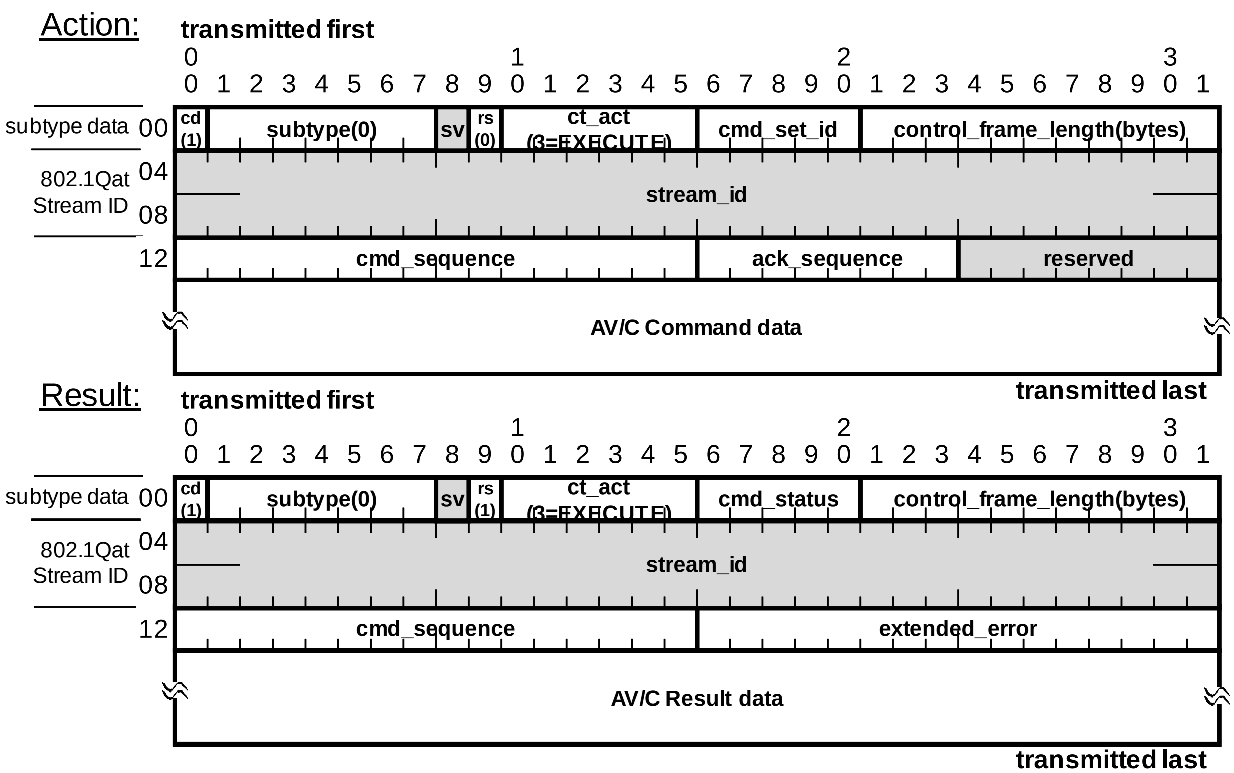
SEQ_RST Result Notes

- **stream_id** is not used set to NULL_STREAM_ID
- **cmd_status** is CTP's status from the action, values:
  - **SUCCESS**: no errors, saved responses discarded, commands in process may or may not complete (but no response will be sent), **NextExpectedSequence** set to 1, **LastAcknowlegedSequence** set to 0
  - **NOT_LOGGED_IN**: no connection exists with requesting station

NOTE: If it is also desired to delete all registered events then use the OPEN action instead of the SEQ_RST action.

6.5.4 EXECUTE action and result

Figure 6.9 - Execute action command transport frames



EXECUTE (Execute) Action

EXECUTE Action Notes

- **stream_id** may be used, if not it is set to NULL_STREAM_ID
- **cmd_set_id** specifies the type of command to be executed:
  - 00_{16} for AV/C
  - FE_{16} for vendor unique, 8 bytes at beginning of Additional Command data are an EUI-64 specifying the command set
  - FF_{16} for the empty command (no operation), used for acknowledge only packets (**cmd_sequence** and **CommandData** fields are not used)
  - Other values TBD (cf. Table 8 in 61883-1)
- **cmd_sequence** is the sequence number value identifying this action, must be the same as the Target's **NextExpectedSequence** value
- **ack_sequence** is the most recently sent **cmd_sequence** value for which no subsequent **cmd_sequence** values are awaiting a response
- **CommandData** field contains the command to be executed as defined by the command set definition

EXECUTE Result

EXECUTE Result Notes

- **stream_id** may be used, if not it is set to NULL_STREAM_ID
- **cmd_status** is CTP's status from the action:

- **SUCCESS**: no errors, command completed, sequence numbers updated, saved responses prior to LastAcknowledgedSequence are discarded, this response is saved (if no state change occurred this is optional), **ResponseData** set according to the command set specification
- **NOT_LOGGED_IN**: no connection exists with requesting station, no actions taken
- **NO_RESOURCES**: not enough resources to save results (i.e. busy), try again later
- **PENDING**: command accepted, but not completed, response will be sent when completed, deciseconds to expected completion in **extended_error**, sequence numbers updated, **ResponseData** is set if specified by the command set specification
- **NOT_SUPPORTED**: the specified cmd_set_id is not supported
- **SEQ_NOT_EXPECTED**: the specified cmd_sequence was not in the range of **LastAcknowledgedSequence**+1 to **NextExpectedSequence**, this means an EXECUTE frame was lost and needs to be retransmitted, **extended_error** contains the value of **NextExpectedSequence**
- **EVENT_REGISTERED**: same as **SUCCESS** except an event has been registered and **extended_error** contains the event ID.

6.5.5 EVENT action and result

<<Editor's note: insert figure here for format of EVENT action and result; both action and result need a 16-bit event_ID field and a 16-bit event_action field plus optional additional data in the result format.

EVENT action

EVENT action notes

- **stream_id** is not used and set to NULL_STREAM_ID.
- **event_ID** specifies the event being changed.
- **event_action** specifies the change to the event as shown in the following table.

event_action	**change specified**
0	acknowledge receipt of EVENT result with event_ID and **EVENT_REGISTERED** status
1	cancel registration for event_ID

EVENT result

EVENT result notes

- **stream_id** is not used and set to NULL_STREAM_ID.
- **event_ID** specifies the event being changed.
- **event_action** is the count of the number of **event_ID** events since it was registered.
- **cmd_status** is CTP's status from the action:
  - **SUCCESS**: no errors, event_ID registration has been canceled, **event_action** contains the count of the number of times this event occurred during the registration
  - **NOT_LOGGED_IN**: no connection exists with requesting station, no actions taken
  - **NOT_SUPPORTED**: the value in **event_action** is not specified in the table above
  - **EVENT_REGISTERED**: a registered event with event_ID has occurred, event_action contains the number of times this event has occurred during this registration.

NOTE: An event notification is an EVENT result sent without a prior EVENT action, it will have a status of **EVENT_REGISTERED**. It will expect a response of an EVENT action from the initiator with the **event_action** field set to zero. If the target doesn't receive this in a reasonable time it will resend the event notification, the initiator can distinguish this from a new occurrence of the event by the count in the **event_action** field. All other EVENT result packets sent by the target are in response to an EVENT action with a non-zero **event_action** field.

6.6 Protocol layering

<<Editors note: Text TBD. Purpose of this section is to document specific layering for 61883/IIDC over AVBTP>>

6.7 Session management

<<Editor's note: Text for this section is still cut and paste from AVBTP assumptions document and John Nels Fuller's contribution, needs further editing and design work>>

6.7.1 Overview

The 61883 over AVBTP session management protocols and procedures:

- Shall use LLDP (802.1AB), SRP (802.1Qat) protocols as an integral part of the Session management protocols and procedures.
  - o >> Thin layer, renaming of Qat service interface
    - >> Specify how to define bandwidth based on applications needs and pass to Qat as TSPEC
    - Reservation will require minimum Ethernet payload size for packet bandwidth reservation calculation, so 1722 or Qat will have to round up for filling of minimum size Ethernet Frame

- Shall provide a service interface to protocols such as Zeroconf
  - o >> Informative annex??

- Shall adapt 1394 AV/C Function Control Protocol (FCP) for use in 61883 over AVBTP.

6.7.2 General Protocol operation

<< Editor's note: The structure for section 0 needs some work, to start with here are the consensus items from our teleconference meeting so far:

- **Function control protocol equivalent is IN**
  - o **AV/C will just be the first command set supported**
  - o **Intention is to not carry 1394 bus resets (use 1394.1 model)**
- **Plug control registers are OUT**
  - o >> Will be part of work being done in 802.1Qat.
- **Connection management procedures are IN**
  - o >>Should be a "thin" layer that maps to 802.1Qat
- **Stream ID Assignment is IN if not defined in 802.1**
  - o **Needed to complete our "Plugs" and CMP**
- **IRM emulation is OUT**
  - o **>> Will be done in 802.1Qat**
- **Service Discovery is IN for each command set supported**
  - o **(i.e. AV/C will recommend Bonjour, but other protocols will be allowed).**
  - o **>>Informative Annex**
- **AVBTP session management will use a Talker-Controller-Listener model for 61883 subtype protocol.**
  - o **>>Controller is part of the model but not part of 1722 (in application layer).**

>>

6.8 Command Transport Protocol (CTP)

This section defines the operation of the Command Transport Protocol (CTP). This protocol is designed to be able to transport the AV/C protocol as used in IEEE 1394 networks, but is extensible to allow for other future protocols.

The Command Transport Protocol (CTP) is used to transport media specific commands between an *initiator* and a *target*. The exact nature of the media specific commands carried by the CTP protocol is beyond the scope of this specification. For details of the format of the CTP message see 6.5.

6.8.1 *results* and *actions*

In order to avoid confusion between CTP packets sent between AVBTP end stations and the commands and responses that they carry, the CTP packets sent by an initiator are called *actions* and the CTP packets sent by a target are called *results*.

6.8.2 CTP operation

6.8.2.1 Handling possible control frame loss

Since many IEEE 802 media do not provide an acknowledged delivery service, CTP accounts for the possibility that actions or results may be lost in transit with no error indication. To do this initiators shall save each EXECUTE action they have sent until the corresponding EXECUTE result is received; while targets shall save each EXECUTE result they have sent until it has been acknowledged by the initiator.

NOTE: An exception is made if the EXECUTE action received by the target represents a command that never changes the state of the device. In this case the target may choose not to save the EXECUTE result and to instead execute the EXECUTE action again when the initiator requests the result to be resent.

This is managed with the sequence numbers in the **cmd_sequence** and **ack_sequence** fields of the EXECUTE action and EXECUTE result packets and with variables **NextExpectedSequence** and **LastAcknowlegedSequence**. When the target receives and EXECUTE action it checks the **cmd_sequence**, if it is in the range **LastAcknowlegedSequence**+1 to **NextExpectedSequence-**1 it is a request to resend a saved EXECUTE result; if **cmd_sequence** equals **NextExpectedSequence** then it is a new action to be executed and **ack_sequence** is used to update **LaskAcknowlegedSequence** (if there are resources to store another EXECUTE result then **NextExpectedSequence** is incremented, otherwise an EXECUTE result with **NO_RESOURCES** status is sent); otherwise an EXECUTE result with **SEQ_NOT_EXPECTED** status is sent.

6.8.2.2 Handling of SEQ_NOT_EXPECTED

When the initiator receives an EXECUTE result with a **SEQ_NOT_EXPECTED** status it knows that previously sent actions have not been received by the target. When this happens or when an EXECUTE result has not been received after a reasonable amount of time has passed since sending an EXECUTE action the initiator resends its saved EXECUTE action unchanged.

6.8.2.3 Handling of NO_RESOURCES

When the initiator receives an EXECUTE result with **NO_RESOURCES** status it must hold the EXECUTE action until it can change it **ack_sequence** to release resources in the target (this is the only condition when the initiator my resend an EXECUTE action that isn't identical to the previously sent EXECUTE action with the same **cmd_sequence**).

6.8.2.4 Establishing a connection with the target (OPEN)

Before an initiator can begin sending commands to a target it must first establish a connection with that target. This is done with the OPEN action and result. If the initiator fails to receive a OPEN result in a reasonable time it may send the OPEN action again (the number of times the initiator will resend the OPEN after a timeout should be at least one).

The initiator shall not send any other actions to that target until it receives a OPEN result with either **SUCCESS** or **ALREADY_LOGGED_IN** status. The target shall not send a OPEN result with **SUCCESS** status unless it can guarantee that it can always store at least one EXECUTE result for this initiator until receipt of the result is acknowledged by the initiator.

6.8.2.5 Disconnecting with the target (CLOSE)

When an initiator is done with a connection to a target it sends a CLOSE action to release the target’s resources. If after a reasonable time the initiator has not received a CLOSE result from the target it sends the CLOSE action again (the number of times the initiator will resend the CLOSE action after a timeout shall be at least one).

<<Editor’s question: Do we need to define what is a “reasonable amount of time”>>

6.8.2.6 Reinitializing the connection with the target

If the initiator has reached its retry limit is sending an EXECUTE action it may need to reinitialize the connection with the target. This is done by sending the SEQ_RST action which tells the target to release all saved responses. If any commands are being executed they may or may not complete but the results will not be sent, **NextExpectedSequence** is set to one and **LastAcknowlegedSequence** is set to zero, and then the target sends the SEQ_RST result. If after a reasonable time the initiator has not received the SEQ_RST result from the target it sends the SEQ_RST action again (the number of times the initiator will resend the SEQ_RST action should be at least one).

Some command sets provide a means for the target to notify the initiator of certain events such as incoming call or a button press. In all known cases the initiator must first issue a command that requests such notification. When this command is delivered to the target and successfully executed (via the EXECUTE action) the status in the result will be EVENT_REGISTERED with a 16-bit event_ID in the extended_error field. Later when the event occurs the target will send an EVENT result (unprovoked by and EVENT action) to the initiator with that same event_ID. The initiator must acknowledge the event by sending and EVENT action to the target with that same event_ID and event_action set to zero. If after a reasonable time the target has not received this acknowledgement and the event registration hasn’t been canceled it will sent the EVENT result again (the number of times the target resends the EVENT result after a timeout shall be at least one).

6.9 Timing and Synchronization

6.9.1 General

Timing and synchronization for IEC 61883 over AVBTP is generally accomplished in the same manner as specified in 61883-1 through 61883-8 over IEEE 1394. The difference is the use of the **avbtp_timestamp** instead of the **SYT** field in the CIP header or the Timestamp field in the Source Packet header. The **SYT** and Timestamp fields may contain valid or invalid data and therefore must be ignored on receipt in an AVBTP node.

Usage of the timing and synchronization information included in the CIP header is generally consistent with the definition in the 61883 series of standards where the main differences are:

1) **The SYT field function formatted in seconds, cycle and cycle offset is replaced by the AVBTP timestamp field expressed in 802.1AS based nanoseconds..**
2) **For protocols of 61883-4 and 61883-7 where the SPH field is set to one, then the 32 bit source packet header field for each source packet contains the associated presentation time of the source packet expressed in IEEE 802.1AS based nanoseconds instead of IEEE 1394 based seconds, cycle and cycle offsets.**
3) **In the IEC 61883 series of specification the term "receiver" is used, whereas in this specification the term "listener" is used.**
4) **In the IEC 61883 series of specification the term "transmitter" is used, whereas in this specification the term "talker"is used.**
5) **IEEE 1394 on networks faster than 100 megabits per second allows CIP packets larger than can fit in a standard single IEEE 802.3 Ethernet frame. This standard does not, if an application needs to send more than data than can fit in a single Ethernet frame, it must generate multiple correctly formatted CIP packets each one that can fit in an 802.3 Ethernet frame.**
6) **Some of the IEC 61883 series of specification sometimes have options that differentiate "professional", "consumer-use" or "cost-sensitive" equipment. This standard does not, and if a device supports a given format, it must support all mandatory requirements as specified in this document.**
7) **Not all formats and their associated protocol technologies, parameters, methods, etc. specified in the 61883 series of specification are supported by this specification (example 61883-6 allows for packed formatted audio data, this specification does not).**

6.9.2 61883-2 timing and synchronization

61883-2 formatted frames shall follow the same timing and synchronization rules as defined in IEC 61883-2 section 6, but using IEEE 802.1AS time in the AVBTP timestamp frame field instead of IEEE 1394 based cycle time in the SYT field. AVBTP talkers shall set the CIP header SYT field to all ones for all transmitted CIP packets and AVBTP listeners shall ignore the SYT field.

The talker shall transmit a valid time stamp value in the AVBTP timestamp field once every video frame period. The time stamp shall be transmitted in an AVBTP frame that meets the following conditions:

- packet_arrival_time_L ≤ time stamp value
- time stamp value – transmission_delay_limit ≤packet_arrival_time_F

where:

packet_arrival_time_F is the IEEE 802.1AS time when the first bit of the packet which has the time stamp has arrived at the listener;

packet_arrival_time_L is the IEEE 802.1AS time when the last bit of the packet which has the time stamp has arrived at the listener;

transmission_delay_limit = default value of 2,000,000 nanoseconds (2 milliseconds).

In case of *Hx* (*H* = 1,2,4) transmission, *KH* data blocks are transmitted in a video frame period *M* using *K* isochronous packets. Isochronous packet n contains *H* data blocks of n*H*, n*H*+1, ... and (n+1)*H*–1.

The isochronous packet n of a video frame period M shall be transmitted on the following conditions (n = 0, ... ,*K*–1):

- packet_arrival_time_L ≤ nominal timing for isochronous packet n

- nominal timing for isochronous packet n – transmission_delay_limit ≤ packet_arrival_time_F

where

packet_arrival_time_F is the cycle time when the first bit of the isochronous packet n has arrived at the receiver;

packet_arrival_time_L is the cycle time when the last bit of the isochronous packet n has arrived at the receiver;

K is the number of isochronous packets without empty packets in a video frame period.
K = 250 (525-60 system)
K = 300 (625-50 system)

Nominal timing for isochronous packet n = $T_M + (T_{M+1} - T_M) \times n/K$

T_M is the time stamp for video frame period M transmitted in the AVBTP timestamp field.

6.9.3 61883-4 timing and synchronization

The timing and synchronization of 61883-4 over AVBTP uses the same method as used in 61883-4 over IEEE 1394 networks in that the source packet header quadlet always contains a valid timestamp. In 61883-4 over AVBTP, this quadlet contains nanosecond 802.1AS based time instead of IEEE 1394 seconds, cycle and cycle offset time.

For 61883-4 formatted frames over AVBTP, section 4.3 of 61883-4 is replaced with the following:

The time stamp in the source packet header is used by listeners for reconstructing a correct timing of the TSPs at their output. The time stamp indicates the intended delivery time of the first bit/byte of the TSP from the listener output to the transport stream target decoder. The time stamp represents the 32 least significant bits of the IEEE 802.1AS at the talker at the moment the first bit/byte of the TSP arrives from the application, plus some offset. The offset is equal to the overall delay of the TSP between the moment of arriving (of the first bit) and the moment the TSP (first bit) is delivered by the receiver to the application. The default value of this offset is equal to 2,000,000 nanoseconds (2 milliseconds).

6.9.4 61883-6 timing and synchronization

<<Editors Note: packed AM24 and flow based rate control will not supported by AVBTP end stations>>

Replace Section 7.2 of 61883-6 with:

In the case where a CIP packet contains multiple data blocks, it is necessary to specify which data block of the CIP corresponds to the IEEE 802.1AS based AVBTP time stamp.

The talker prepares the time stamp for the data block which meets this condition:

mod(data block count, SYT_INTERVAL) = 0 (1)

where

- data block count is the running count of transmitted data blocks;
- SYT_INTERVAL denotes the number of data blocks between two successive valid AVBTP timestamps which includes one of the data blocks with a valid AVBTP timestamp. For example, if there are three data blocks between two valid AVBTP timestamps, then the SYT_INTERVAL would be 4.

The listener can derive the index value from the DBC field of a CIP with a valid AVBTP timestamp using the following formula:

index = mod((SYT_INTERVAL – mod(DBC, SYT_INTERVAL)), SYT_INTERVAL) (2)

where

- index is the sequence number;
- SYT_INTERVAL denotes the number of data blocks between two successive valid AVBTP timestamps, which includes one of the data blocks with a valid AVBTP timestamp;

- DBC is the data block count field of a CIP.

The listener is responsible for estimating the timing of data blocks between valid time stamps.

The method of timing estimation is implementation-dependent.

Replace Section 7.3 of 61883-6 with:

A data block contains all data arriving at the talker within an audio sample period. The data block contains all the data which make up an “event”.

The talker shall specify the presentation time of the event at the receiver. A receiver for shall have the capability of presenting events at the time specified by the transmitter.

If a function block receives a CIP, processes it and subsequently re-transmits it, the AVBTP timestamp of the outgoing CIP shall be the sum of the incoming AVBTP timestamp and the processing delay.

The transmitter shall add TRANSFER_DELAY to the quantized timing of an event to construct the AVBTP timestamp. The TRANSER_DELAY value is initialized with the DEFAULT_TRANSFER_DELAY value. Note that for all talkers, the TRANSFER_DELAY may be changed to achieve a shorter TRANSFER_DELAY value to allow for a shorter time if the end to end delay in the AVB network can allow it.

The DEFAULT_TRANSFER_DELAY value is 2,000,000 nanoseconds (2 milliseconds)

6.9.5 61883-7 timing and synchronization

Replace IEC 61883-7 section 5.1.3 with:

The source packet header field is a one quadlet field (4 bytes) that represent an 802.1AS based time stamp.

The time stamp is used by 61883-7 capable AVBTP listeners for reconstructing a correct timing of the transport stream packets at their output. The time stamp indicates the intended delivery time of the first bit/byte of the transport stream packets from the receiver output to the T-STD (Transport Stream Target Decoder). The time stamp represents the least significnate 32 bit binary time of the IEEE 802.1AS based clock at the moment the first bit/byte of the transport stream packet arrives from the application, plus an offset which is equal to the overall delay of the transport stream packet between the moment of arriving (of the first bit) and the momentthe transport stream packet (first bit) is delivered by the receiver to the application.

The default value of this offset shall be 2,000,000 nanoseconds (2 milliseconds).

6.9.6 61883-8 timing and synchronization

<<Editor's note: TBD>>

6.10 Service Interface

<<Editor’s note: TBD>>

7. Proprietary/Experimental subtype AVBTP protocol

<<Editor's note: introductory text TBD.>>

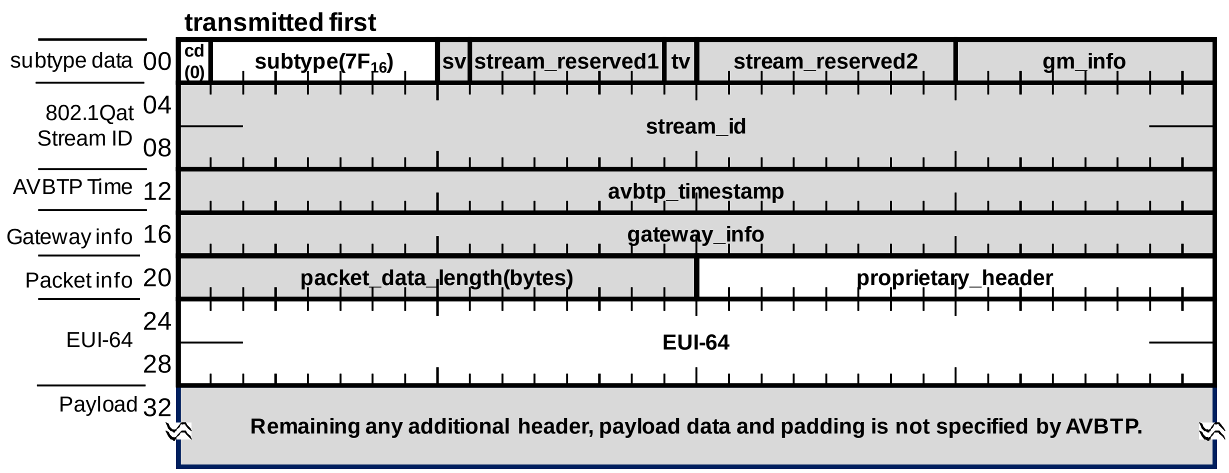
7.1 Overview

<<Editor's note: overview text TBD>>

7.2 Proprietary/Experimental subtype stream data format

<<Editor's note: section text TBD>>

Figure 7.1 -- Proprietary/Experimental stream data format



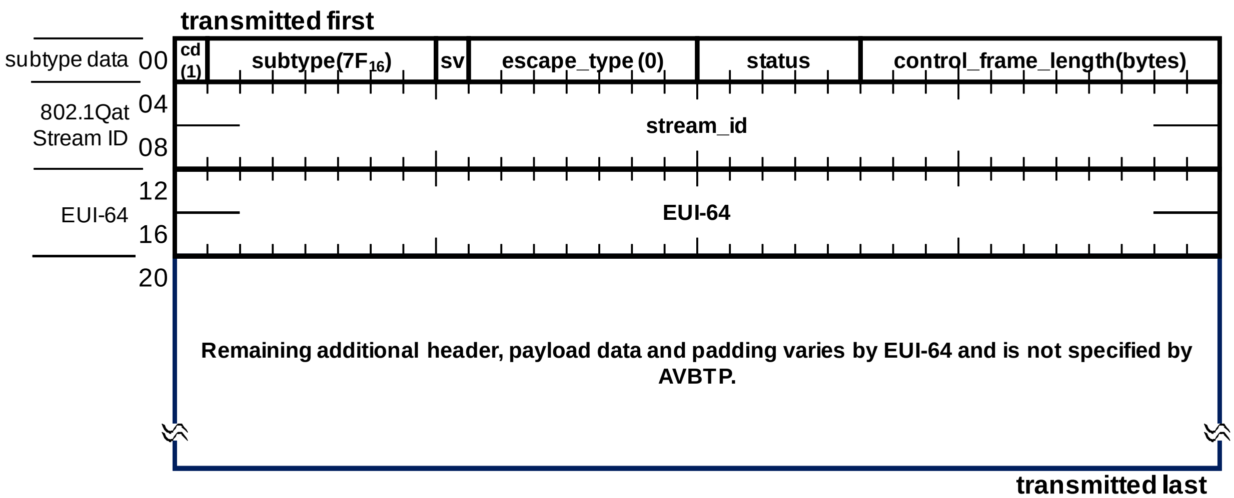
7.3 Proprietary/Experimental subtype control format

For subtype $7F_{16}$ AVBTP control frames the subtype_data1 field is called the escape_type field and has the following values:

a) A value of zero (0) in the subtype_data field is required for use by proprietary/experimental as this field

b) Values of 1 through 255 are reserved for future use by this standard.

Figure 7.2—Proprietary/Experimental control escape subtype format



<<Editor's note: need to add "as required" padding to ensure that the last quadlet is filled if the length does not fill the last quadlet in the frame>>

If the subtype is $7F_{16}$ and the subtype_data field is 0, then following the subtype_data field shall be a unique EUI-64 field that identifies the proprietary/experimental protocol.

All data after the EUI-64 is available for use by the proprietary/experimental protocol and is beyond the scope of this specification.

Annexes

Annex A (informative) Bibliography

[B1] IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition.

[B2] IEEE EUI-64, IEEE EUI-48, and IEEE MAC-48 assigned numbers may be obtained from the IEEE Registration Authority, http://standards.ieee.org/regauth/. Tutorials on these assigned numbers may be found on this web site.

[B3] Digital Transmission Content Protection Specification Volume 1 (Informational Version)

[B4] DTCP Volume 1 Supplement D DTCP use of IEEE1394 Similar Transports (Informational Version)

THIS PAGE LEFT INTENTIONALLY BLANK

<<Editor's note: This is because the template automatically forces Annexes to start on an odd numbered page>>

Annex B (normative) Interworking 61883 between AVBTP and IEEE 1394 networks

B.1 Introduction

<<Editor’s note: Text TBD>>

B.2 1394 to/from IEEE 802 AVBTP Interworking scenarios

<< Editor’s note: To kickoff work on this section, I’ve copied and edited excerpts from a previous email with “use case” scenarios that I believe we need to handle:

Note: IWU below is an acronym for "Interworking Unit" (instead of using the term "bridge", which I prefer that term for our discussions as to me the term "bridge" usually implies less processing and also if we keep saying 1394 bridge, then you have to deal with the asynchronous and control traffic as well).

In the case where you have a stream originated by a 1394 device on a 1394 bridged network, and you want to have the scenario "A" of:

```
1394 Talker -> 1394 Net- IWU1 -> AVB Net -> IWU2 -> 1394 Net -> 1394 Listener(s)
```

I would also assume that for that case, one could also have AVB Listeners on the same stream as well as well with the following scenario "B":

```
1394 Talker -> 1394 Net1- IWU1 -> AVB Net -> IWU2 -> 1394 Net2 -> 1394 Listener(s)
                                   |
                                   V
                             AVBTP Listener(s)
```

I would again assume we want to support AVB talkers with streams going to 1394 listeners, scenario C:

```
AVB Talker -> AVB Net -> IWU2 -> 1394 net2 -> 1394 Listener(s)
```

I also think that whatever mechanism we come up with supports all three scenarios and I am wary of a case that excludes any of them. Based on my understanding (I could be wrong here) that you proposal to have a different AVBTP timestamp meaning makes case B not happen and that would exclude AVBTP devices to join in the stream.

To support these cases, AVBTP Interworking Units will have to handle timing and synchronization conversion/adaptation. This will have to be a work item in addition to session control, stream reservation, fragmentation, etc. Based on that I’ve put together an outline for the rest of the section. Additional comments and suggestions are welcome

>>

B.3 IWU session management

<<Editor’s note: TBD>>

B.3.1 General

<<Editor’s note: TBD>>

B.3.2 Stream join

<<Editor's note: TBD>>

B.3.3 Data transfer

<<Editor's note: TBD>>

B.3.4 Stream leave

<<Editor's note: TBD>>

B.3.5 IWU data adaptation

<<Editor's note: TBD>>

B.3.6 General

<<Editor's note: TBD>>

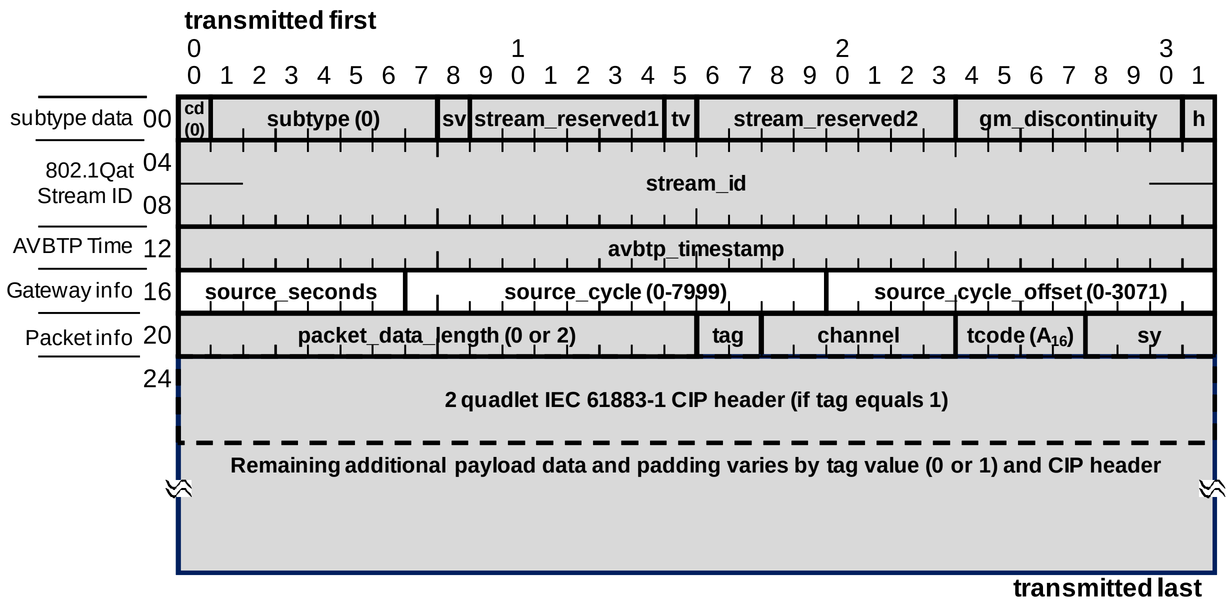
B.3.7 Encapsulation

IEEE 1394 to AVBTP interworking units shall use the same format of the AVBTP IIDC/61883 as specified in clause 6 above.

IIDC/61883 data stream frames shall use the **gateway_info** field as specified in 5.6.7 above. This section defines how that field is formatted for IEEE 1394 to AVBTP interworking units.

For IEEE 1394 to AVBTP interworking units the gateway_info is formatted into the following fields:

a) **source_seconds**: most significant 7 bits of the quadlet
b) **source_cycle**: 13 bits with a value range of 0-7999 decimal indicating the source 1394 bus cycle of the frame
c) **source_cycle_offset**: 12 bits with a value range of 0-3071 indicating the source 1394 bus cycle offset of the frame


transmitted first
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
subtype data 00
cd (0)
subtype (0)
sv
stream_reserved1
tv
stream_reserved2
gm_discontinuity
h
802.1Qat Stream ID 04 08
stream_id
AVBTP Time 12
avbtp_timestamp
Gateway info 16
source_seconds
source_cycle (0-7999)
source_cycle_offset (0-3071)
Packet info 20
packet_data_length (0 or 2)
tag
channel
tcode (A16)
sy
24
2 quadlet IEC 61883-1 CIP header (if tag equals 1)
Remaining additional payload data and padding varies by tag value (0 or 1) and CIP header
transmitted last

B.3.8 Stream Sequencing, Fragmentation and Reassembly procedures

CIP packets on a 1394 network may be larger than can be accommodated on the AVBTP network which has a maximum AVBTP frame size (header plus payload) of 1492 bytes.

If a 1394 to Ethernet AVBTP gateway receives a CIP packet that will not fit on the AVBTP network, then the gateway shall break up the large CIP packet into multiple smaller CIP packets prior to sending on the AVBTP network.

If an Ethernet AVBTP to IEEE 1394 gateway receives multiple packets for a given cycle and that 1394 network can support large CIP packets, then the 1394 gateway shall reassemble the smaller AVBTP based CIP packets into larger IEEE 1394 based CIP packets as necessary.

B.4 Timing and synchronization

<<Editor’s note: TBD. This section is intended to list any specific requirements of gateways to map timing and synchronization field (e.g. mapping of SYT fields to and from AVBTP timestamps). Below are some notes that still appear useful from previous work we’ve done>>

1394 to 1722 conversion:

a) **-** Convert SYT field to AVBTP presentation time

b) Leave SYT field intact – AVB ignores it

c) Exchange cross-timestamp packets with other 1394/AVB Gateways

d) Could strip the 32-bit SPH to save a quadlet

  1) **Not really worth while**

  2) **Would introduce jitter on 1394-to-AVB-to-1394**

  3) **AVB Listener ignores SYT field**

1722 to 1394 conversion:

a) If SID=63 (AVB Talker)

  1) **Convert AVB Presentation Time to SYT field**

  2) **Possible problems with 2ms SYT field on Part 2, 3, 5 & 6**

b) Exchange cross-timestamp packets with other 1394/AVB Gateways

c) Possibly recreate SPH if 1394-to-AVB gateway stripped it when putting 1394 packet onto the AVB network

  1) **If SID <> 63**

  2) **And SPH = 1 (MPEG traffic, IEC 61883-4 and 61883-7)**

d) Larger range of AVB Presentation Time Offsets could require buffering in gateway

B.4.1 General

<<Editor’s note: TBD>>

B.4.2 Timing adaptations/control

<<Editor’s note: TBD>>

B.4.3 Synchronization

<<Editor's note: TBD>>

B.4.4 Queuing/Scheduling Mechanisms

<<Editor's note: TBD>>

THIS PAGE LEFT INTENTIONALLY BLANK

<<Editor's note: This is because the template automatically forces Annexes to start on an odd numbered page>>

Annex C(normative) MAC Address Acquisition Protocol (MAAP)

C.1 Overview

<<Editor’s note: I have significantly changed this section based on verbal feedback from AVBTP and AVB meetings, so the intent is not to have this section perfect, but hopefully with enough detail to start the conversation on the changes that I was OK’d to take a first cut at.

In general, more and more this protocol is being discussed as a possible method to allocate not just Multicast addresses for AVBTP, but also for other 802.1 uses where multiple unique multicast and unicast addresses (such as VMWare, applications that need guaranteed unique user assigned source MAC addresses, IEEE 1733, etc.). Also I have been tasked to do on this first cut:

1) **Make it so we can have shared Multicast MAC addresses in the future**
    - i) **I have not put in any text about this yet, but my thought is to use the new flags field and create a shared address indicator where the application can request a Multicast MAC address range that either can or cannot be shared. Comments on this would be appreciated**
    - ii) **<<F2F notes: Not our job. That should be done at another layer. Want to keep the layer as thin as possible.>>**
2) **Allow the protocol to allocate both Multicast and Unicast addresses**
3) **Allow the protocol to accommodate a future MAC address allocation server (similar to DHCP for IPv4) where you are told to use a specified address as given by a server rather than a random selection (this is also done to speed up the process if a server is available)**
    - i) **With that, I’ve added ASSIGN and UNASSIGN messages so that a server can say “I know you asked for range A, but I want/need you to use range B instead”**

Another thing I’ve done is gone back and read Dr. Stuart Cheshire’s Zeroconf book in that I wasn’t happy with the first draft where you’d have to wait for 10 seconds to get a MAC address. Based on reading Zeroconf’s methods for both IPv4 and DNS name assignment, I’ve added an announce phase (used by IPv4 link local assignment) and also used shorter timeouts (used by name assignment).

All comments and questions are welcome.>>.

<<Phone and F2F meeting notes:

Phone notes:

gm_info (fixed source slides)

discuss multicast address

Change to optional on tag. Recommended.

receive shall process accept both, recommended on transmit

8 bits lsb

lease time

>>

Multicast MAC addresses will be required by AVBTP for the transmission of media stream from one talker to multiple listeners. As AVBTP runs directly on a layer 2 transport, there is no existing protocol to dynamically allocate multicast MAC addresses for AVBTP.

A reserved block of Multicast MAC addresses has been reserved for the use of AVBTP these are from xx-xx-xx-xx-xx-xx through yy-yy-yy-yy-yy-yy. **<<Editor’s note, this range will be filled in once we have the range allocated to us from the IEEE>>**

The MAC address Acquisition Protocol (MAAP) specifies a mechanism to dynamically allocate Multicast and Unicast MAC addresses in a specified address range. The base protocol uses a request, announce, defend and release mechanism and also has been designed to allow for future enhancements of supporting a MAC address assignment server via the assign and un-assign mechanisms.

In the case of AVBTP using MAAP, any application that uses addresses from the AVBTP multicast address range must make use of the AVBTP Multicast Acquisition Protocol to request and defend those addresses.

To obtain a set of addresses, an application randomly selects a multicast address from the desired range and multicasts probe packets containing the desired address range. The application then listens for a defend response or assign command. A defend response indicates that the address is in use. If no responses are received the multicast address is then assigned for use by the application. If any response is received indicating the address is already in use, then the MAAP layer randomly selects a new address range and begins sending probe packets containing the new address range. The process is repeated until an address range is successfully obtained at which point the MAAP layer informs the application of the resulting address range.

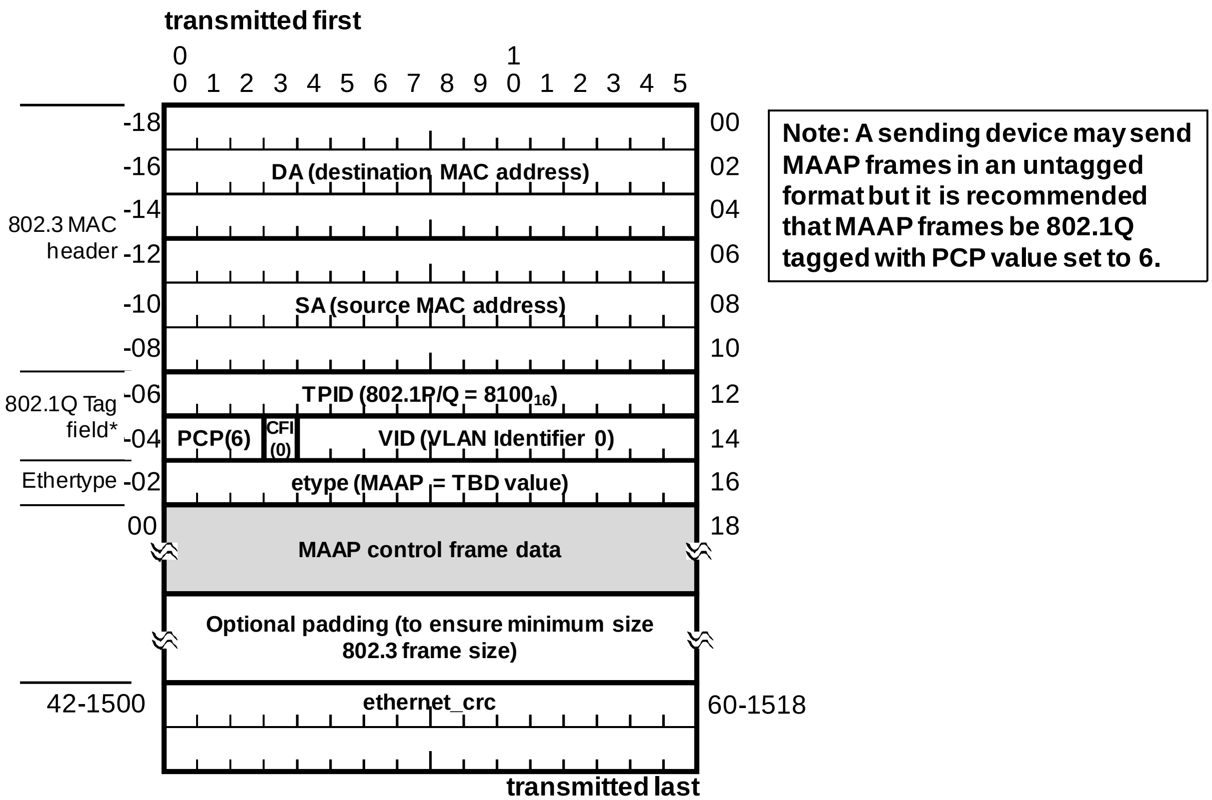
Once the MAAP layer has obtained an address it is required to remember and defend the address until it is until it is released by the application. Defending an address consists of listening for probes to use the address and responding to the probes signifying that the address is already in use.

MAC addresses can be allocated individually or as a consecutive range. A MAAP entity that is already acquired any address in a probed range must respond to the probes to defend its address.

C.2 Protocol Message Format

All MAAP frames can be transmitted with or without an 802.1Q tag in the frame. It is recommended that MAAP frames be transmitted using 802.1Q tagging. If the frame is tagged, then it shall be transmitted using 802.1Q field with a fixed Priority value of 6 and a VLAN ID of zero(0). For 802.3 and example is shown in the figure below.

Figure C.1 – 802.3 MAAP frame format

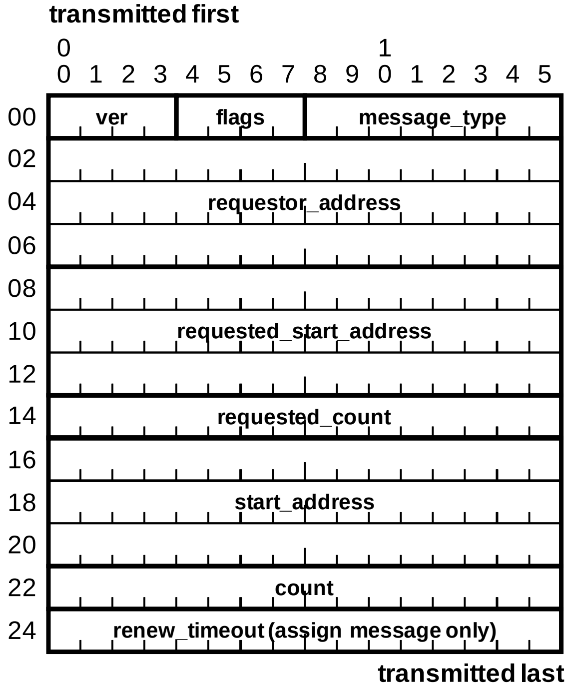


The MAAP control frame data consists of the following fields in the following order:

a) **version_and_flags:** field: 8 bit byte with the following subfields defined:
   1) **ver (version) most significant 4 bits of this byte**
   2) **flags: least significant 4 bits of this byte**
b) **message_type**: 1 byte
c) **requestor_address**: 6 bytes
d) **requested_start_address**: 6 bytes
e) **requested_count**: 2 bytes
f) **start_address**: 6 bytes
g) **count**: 2 bytes
h) **renew_timeout**: 2 bytes

A diagram of the MAAP frame format is shown in the following figure

Figure C.2 – 802.3 MAAP frame format



. The frame type is identified in the message_type field as follows:

Table C.1—MAC address Acquisition protocol message types

Value (decimal)	FUNCTION	Meaning
0	--	Reserved
1	MAAP_PROBE	Probe MAC address(es) frame
2	MAAP_DEFEND	Defend address(es) response frame
3	MAAP_ANNOUNCE	Announce MAC address(es) acquired frame
4	MAAP_RELEASE	Release MAC acquired address(es)
5	MAAP_ASSIGN	Assign MAC address(es) command
6	MAAP_UNASSIGN	Unassign MAC address(es) command

All MAAP frames are sent with a multicast destination MAC address set to the reserved MAAP protocol Multicast address of zz-zz-zz-zz-zz-zz. **<<Editor's note, this reserved Multicast address shall be set in a future revision of this document per IEEE rules and procedures. Current suggestion is that it is in the same OUI and address range of the Multicast Registration Protocol (MRP)>>**

The Source MAC Address shall be set to the MAC address of the sender.

If the MAAP frame is 802.1Q tagged then:

The first Ethertype field shall be set to the 802.1Q Ethertype of $81\text{-}00_{16}$.

The Priority Control Point (PCP) field shall be set to 6.

The Canonical Format Indicator shall be set to zero(0).

The VLAN Identifier shall be set to zero(0).

The main Ethertype field shall be the MAAP reserved Ethertype of nn-nn_{16}.**<<Editor's note: This Ethertype will be allocated in a future version of this specification per normal IEEE procedures>>**

In a Probe message, the requested_start_address and requested_count fields shall be set to the start MAC addresses of the requested range and the total number of MAC addresses requested.

The requested_start_address and requested_count represent an inclusive request for all MAC addresses between a range of addresses.

<<Editor's note: I was requested at the last face to face meeting to "use the text from MMRP" to describe how MAC addresses are to be documented. In looking at the current spec and its corrigendum, I did not see which text to use as what I saw in the spec seemed to not describe the value and range, but instead "punted" on the subject. Please advise exactly which text I should use and if it needs to be modified to meet the requirements of this specification.>>

In a Response message, the Start Address and count fields shall be set to values of the start and count value of the range of addresses that conflict with the requested range

C.3 Requesting an address range

An application allocates an address by requesting the MAAP entity to acquire a specified number of Multicast or Unicast addresses.

If the application has previously obtained an address range and has access to persistent storage, the application should have recorded the previous address range and should attempt to reuse the saved address range.

If no previous addresses were previously allocated then the MAAP entity randomly selects an address range based on the number of addresses requested by the application to acquire and a range of addresses specified by the application (for example, the AVBTP reserved multicast address range) . The MAAP entity selects a subset of that range based on the count of addresses requested and where the requested start address is selected using a pseudo-random number generator with a uniform distribution across the reserved range xx-xx-xx-xx-xx-xx through yy-yy-yy-yy-yy-yy.

The pseudo-random number generation algorithm must be chosen so that different hosts do not generate the same sequence of numbers for subsequent Probe frames. The pseudo-random number generator should be seeded using the least significant bytes of IEEE 802 MAC address of the requestor.

Once the address is selected the MAAP entity will start the address acquisition process. It starts by:

- Setting the maap_probe_counter to MAAP_PROBE_RETRANSMITS.
- Formatting a new probe frame with:
    - requestor_start_address field set to the Unicast MAC address associated with the MAAP entity
    - requested_start_address field set to the start of the address range requested
    - requested_count field set to the number of contiguous MAC addresses requested
    - start_address field set to 00:00:00:00:00:00
    - count field set to zero
- Sending the frame to the network
- Decrementing the maap_probe_counter by 1
- Starting the maap_probe_timer setting it to a random value selected uniformly in the range between MAAP_PROBE_INTERVAL_BASE to MAAP_PROBE_INTERVAL_BASE plus MAAP_PROBE_INTERVAL_VARIATION milliseconds,

If a defend response or announce indication frame is received that contains a conflicting address range as reported in the start_address and count of that frame, then the MAAP entity shall randomly select a new set of addresses, and set the maap_probe_counter to MAAP_PROBE_RETRANSMITS..

If an assign message is received where the requestor_address, requested_start_address and requested_count match the sent probe message, then the MAAP entity will start the announce process and inform the application of the assigned MAC address range from the start_address and count fields of the incoming message.

If a maap_probe_timer expires, then the MAAP entity will decrement the maap_probe_counter.

- If the maap_probe_counter is equal to zero, then the MAAP entity will inform the application that the Address range has been acquired and the MAAP entity will start the announce process.
- If the maap_probe_counter is greater than zero, then the MAAP entity will retransmit the current address range, decrement the maap_probe_counter by one and restart the maap_probe_timer to a new random value selected from uniformly in the range between MAAP_PROBE_INTERVAL_BASE to MAAP_PROBE_INTERVAL_BASE plus MAAP_PROBE_INTERVAL_VARIATION milliseconds,

C.4 Announcing an acquired MAC Address Range

Once an address range is acquired, the local MAAP entity shall announce to the network by sending announce messages to the network. It does this by:

- Setting the maap_announce_counter to MAAP_ANNOUNCE_RETRANSMITS.
- Formatting a new announce frame with:
  - requestor_start_address field set to Unicast MAC address associated with the MAAP entity
  - requested_start_address field set to the start of the address range acquired
  - requested_count field set to the number of addresses acquired
  - start_address field set to the start of the address range acquired
  - count field set to the number of addresses acquired
- Sending the frame to the network
- Decrementing the maap_announce_counter by 1
- Starting the maap_probe_timer setting it to a random value selected uniformly in the range between MAAP_ANNOUNCE_INTERVAL_BASE to MAAP_ANNOUNCE_INTERVAL_BASE plus MAAP_ANNOUNCE_INTERVAL_VARIATION milliseconds,

<<Editor's note: need more text here about how announces repeat until count is zero, etc. and have similar text or refer to C.5 on defending and address during the announcement phase. >>

C.5 Defending a MAC Address Range

Once the MAAP entity has acquired a set of addresses it must also defend those addresses.

If the MAAP entity receives a Probe that conflicts with any of its acquired addresses if shall send a Defend response frame back to the source of the Probe. The Response frame shall contain copies of the requestor_address, requested_start_address and requested_count from the received Probe frame. It also then reports start and count of the conflicting address range in the start_address and count fields of the defend response frame. In the case that the application has obtained multiple MAC address ranges that conflict with the request, then any one of the address ranges that conflict shall be sent in the response. If another Request frame is received that conflicts with the address range not sent in the response, then a response frame shall be sent back to the source containing this address range.

The MAAP entity may send multiple Response packets to a received Probe if the application has multiple address ranges that conflict with the address range specified in the probe.

If the MAAP entity receives an Announce message that conflicts with any of its assigned MAC addresses, then another MAAP entity has a conflicting address range that it has acquired (possibly due to message loss, merging networks, etc.). In that case the receiving may send one and only one Defend message for that address range. If a subsequent announce message is sent from that remote MAAP entity, then the other side will not yield, so the local MAAP entity shall relinquish that MAC address range. shall send a Defend response frame back to the source of the Probe and informing the application of the address range being relinquished.

Annex Z(informative) COMMENTARY

This is a temporary Annex intended to record issues/resolutions thereof as the project proceeds. It will be removed prior to Sponsor ballot, and should be ignored for the purposes of TG/WG ballot.

Z.1 Unicast Destination MAC address support for stream data frames?

Will we allow unicast MAC addresses now that we have a stream ID field? Previous versions said that we would use only multicast MAC addresses for stream data frames and that the multicast MAC address would serve as the Stream identifier. Now that we have a 64 bit stream ID in each frame, do we now also allow unicast MAC addresses?

<<Editor's note: Consensus appears to be that AVBTP will not specify this and instead will "point to" 802.1Qat for all management and specification of Stream IDs and their associated MAC addresses.>>

Z.2 Need mechanism for getting PCP value for ClassA and ClassB streams

Group opinion from 2007-09-23 Santa Clara meeting: We should hopefully have some mechanism to make this more plug and play to have some mechanism to automatically have AVB bridges inform AVBTP end stations on what the current value of Priority Code Point (PCP) is to be used for classA and classB streaming traffic if it is changed from the default.

<< Editor's update 2007-10-04: This is currently being discussed as part of the work being done for 802.1Qat.>>

Z.3 Does AV/C protocol need its own subtype?

Would it be more appropriate to make AV/C the standard control for 61883 and use cd of 1 and subtype of 0? Can or should AV/C be used for anything besides 61883/IIDC encapsulation??

Z.4 Need to define more details on format and function of AVBTP source Timestamp and relationship to use of the 61883 SYT field

- Current consensus high level definition is “Nominal launch time (launch to the network)”.
- For fragments, needs to be the same value for each fragment. 61883 type packets will want to be launched in 125us intervals (8 kHz clock)
- AVBTP will also need to deal with time changes in 802.1AS.
- Alternate proposal is that it is not tied to an 8kHz clock and is instead tied to the media clock and can be used as presentation time.
- Format of field is TBD (based on decision from 802.1AS)
- Full resolution target at ~1 second.
- Discussions to date have been where this timestamp indicates the “desired transmit time” for egress frames (i.e. when the frame is scheduled to be passed to 802.1Qav shaping for subsequent egress transmission).
- Further discussions will hopefully clarify the use of this field as we get further in our work on Timing and Synchronization details.

<<Editor’s note: We have made good progress on this based on the work done by Craig Gunther (see current draft sections for changes in timing and synchronization for common, 61883 and Annex B. Current consensus (to my understanding) is:

1) **we will not use the CIP header SYT field for the AVBTP end stations, we will use the AVBTP timestamp field. Only 1394 to 1722 gateways will need to deal with time translations/management.**
2) **we will keep the avbtp_timestamp field at 32 bits**
3) **we will use nanoseconds since epoch giving a resolution of around 4 seconds.**
4) **avbtp_timestamp will be used for presentation time similar to the way that presentation time is used in IEC 61883.**
5) **we will default to 2 milliseconds to add to sample data ingress time to create presentation time to put into the frame.**
6) **for 61883 type traffic with SPH=0 we will use the SYT_INTERVAL mechanism (DBC used to calculate which sample frame the timestamp is related to), but use the avbtp_timestamp field instead of the SYT field.**

With #6 above, we still need to work on how to handle SPH=1 for MPEG 61883-4 and 61883-7 traffic where in 1394/61883, the SYT field is in the source packet and not in the CIP header

>>

Z.5 Need to define how 802.1Qat stream IDs are used by AVBTP

- Need to define relationship of stream IDs with source and destination MAC addresses.
- Need to have a standard Stream ID value that indicates that stream ID contains no data (perhaps all zeros or all ones?) so we can have control frames that either relate to a single stream or to a protocol options for a protocol (subtype). I would assume all zeros would be better as all ones in addresses are usually used for Broadcast.
- Need to define how stream ID management ties into AVBTP session management.

Z.6 Need to define what happens if presentation time has passed and/or is out of range.

<<Editor's note: cut and paste from Craig Gunther's contribution>>

1. What about playing samples if presentation time has already passed?
    - What if it only happens once?
    - What if it consistently happens?
    - Consumer only?
    - Visual indication if samples are discarded?
2. What about presentation time that is so far in the future that the node can't buffer it?

<<END OF CURRENT DRAFT DOCUMENT>>