

INTERNATIONAL IEEE Std 1588™ STANDARD

Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

IECNORM.COM : Click to view the full PDF of IEC 61588:2021



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2019 IEEE

All rights reserved. IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Inc. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the IEC Central Office. Any questions about IEEE copyright should be addressed to the IEEE. Enquiries about obtaining additional rights to this publication and other information requests should be addressed to the IEC or your local IEC member National Committee.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland
Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue
New York, NY 10016-5997
United States of America
stds.info@ieee.org
www.ieee.org

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, ...). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC online collection - oc.iec.ch

Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org

The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 18 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

INTERNATIONAL IEEE Std 1588™ STANDARD

Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.110; 35.240.50

ISBN 978-2-8322-9811-4

Warning! Make sure that you obtained this publication from an authorized distributor.

Contents

1. Scope.....	14
2. Normative references	15
3. Definitions, acronyms, and abbreviations.....	16
3.1 Definitions	16
3.2 Acronyms and abbreviations.....	24
4. Conventions.....	26
4.1 Descriptive lexical form syntax.....	26
4.2 Word usage	27
4.3 Behavioral specification notation	28
5. Data types and on-the-wire formats	30
5.1 General	30
5.2 Primitive data type specifications.....	30
5.3 Derived data type specifications.....	31
5.4 On-the-wire formats	35
6. Clock synchronization model	36
6.1 General requirements on implementations.....	36
6.2 Principal assumptions about the network and implementation recommendations.....	39
6.3 PTP Networks	39
6.4 PTP message classes.....	40
6.5 PTP device types	41
6.6 Synchronization overview.....	53
6.7 PTP communications overview.....	65
7. Characterization of PTP entities	69
7.1 Domains.....	69
7.2 Timescales used in PTP	72
7.3 PTP communications	74
7.4 PTP communication media	81
7.5 PTP Ports	84
7.6 PTP Instance characterization	89
7.7 PTP timing characterization.....	103
8. PTP data sets.....	105
8.1 General specifications for data set members	105
8.2 Data sets for PTP Instances.....	113
8.3 Data sets for Transparent Clocks.....	141
8.4 commonMeanLinkDelayService data sets	144
9. PTP for Ordinary Clocks and Boundary Clocks	144
9.1 General protocol requirements for PTP Ordinary Clocks and Boundary Clocks	144
9.2 State protocol	145
9.3 Best master clock algorithms	154
9.4 Grandmaster PTP Instance timePropertiesDS updates	164
9.5 PTP message processing semantics	166
9.6 Changes in the PTP Instance.....	182
10. PTP for Transparent Clocks.....	182
10.1 Requirements for both end-to-end and peer-to-peer Transparent Clocks.....	182
10.2 End-to-end Transparent Clock requirements.....	182
10.3 Peer-to-peer Transparent Clock requirements	191
11. Clock offset, path delay, residence time, and asymmetry corrections	195

11.1 General specifications.....	195
11.2 Computation of <offsetFromMaster> in Ordinary Clocks and Boundary Clocks	196
11.3 Delay request-response mechanism for Ordinary Clocks and Boundary Clocks.....	197
11.4 Peer-to-peer delay mechanism	199
11.5 MDMI interface and Special Ports	204
12. Synchronization and syntonization of clocks.....	214
12.1 Clock adjustments	214
12.2 Syntonization	214
12.3 Synchronization.....	215
13. PTP message formats	216
13.1 General.....	216
13.2 General PTP message format requirements.....	216
13.3 Header.....	216
13.4 Suffix.....	222
13.5 Announce message	222
13.6 Sync and Delay_Req messages	224
13.7 Follow_Up message	224
13.8 Delay_Resp message	225
13.9 Pdelay_Req message	225
13.10 Pdelay_Resp message	226
13.11 Pdelay_Resp_Follow_Up message.....	226
13.12 Signaling message	227
13.13 PTP management message.....	228
14. TLV entity specifications	228
14.1 General requirements.....	228
14.2 Propagation of TLVs through Boundary Clocks	230
14.3 Vendor and standard organization extension TLVs.....	231
14.4 PAD TLV (optional).....	233
15. PTP management messages (optional)	234
15.1 General.....	234
15.2 PTP management mechanism	234
15.3 Processing of PTP management messages.....	234
15.4 PTP management message format.....	235
15.5 Management TLVs.....	238
16. General optional features.....	272
16.1 Unicast message negotiation (optional)	272
16.2 Path trace (optional).....	280
16.3 Alternate timescale offsets (optional)	282
16.4 Holdover upgrade (optional).....	289
16.5 Isolation of PTP Instances running under profiles specified by different standards organizations (optional).....	290
16.6 Common Mean Link Delay Service (optional).....	291
16.7 Configurable correction of timestamps (optional).....	298
16.8 Calculation of the <delayAsymmetry> for certain media (optional)	299
16.9 Mixed multicast/unicast operation (optional).....	301
16.10 Cumulative frequency transfer method for synchronizing clocks (optional).....	304
16.11 Slave Event Monitoring (optional)	308
16.12 Enhanced synchronization accuracy metrics (optional)	317
16.13 Message Length Extension (optional).....	323
16.14 PTP integrated security mechanism (optional).....	324
17. State configuration options.....	338
17.1 General.....	338
17.2 Grandmaster clusters (optional).....	338

17.3 Alternate master (optional).....	340
17.4 Unicast discovery (optional)	343
17.5 Acceptable master table (optional)	345
17.6 Mechanism for external configuration of a PTP Instance's PTP Port state (optional).....	347
17.7 Reduced state sets and use of the <foreignMasterList> feature (optional).....	353
18. Interactions between PTP Instances in different PTP domains	354
18.1 General specifications.....	354
18.2 Interfaces enabling interdomain interactions.....	355
19. Compatibility of this edition with earlier and future editions	355
19.1 General.....	355
19.2 Compatibility between version 2 and future versions.....	356
19.3 Compatibility with IEEE Std 1588-2002	356
19.4 Compatibility between the PTP Instance conformant to this edition and the implementations conformant to IEEE Std 1588-2008	356
20. Conformance	361
20.1 Conformance objective	361
20.2 PTP conformance requirements.....	361
20.3 PTP Profiles	362
Annex A (informative) Using the Precision Time Protocol (PTP).....	365
A.1 Overview.....	365
A.2 Physical layout	366
A.3 Logical layout.....	366
A.4 Component issues.....	367
A.5 Local implementation issues	368
A.6 System implementation issues.....	370
A.7 Guidelines to achieve optimal performance	371
A.8 Recommendations to aid in conformance testing.....	371
A.9 Recommendation for implementations in unicast networks or networks with non-PTP bridges and routers	372
Annex B (informative) Timescales and epochs in PTP.....	375
B.1 General considerations.....	375
B.2 UTC, TAI and the PTP epoch and timescale updates	375
B.3 Standard time sources	378
B.4 Meaning and uses of the attributes of the timePropertiesDS data set.....	379
Annex C (normative) Transport of PTP over User Datagram Protocol over Internet Protocol Version 4.....	382
C.1 General.....	382
C.2 UDP port numbers.....	382
C.3 IPv4 multicast addresses	382
C.4 sdoId field values.....	383
C.5 Optional values.....	384
C.6 IPv4 Options.....	384
C.7 Protocol addresses	384
Annex D (normative) Transport of PTP over User Datagram Protocol over Internet Protocol Version 6.....	385
D.1 General.....	385
D.2 UDP port numbers.....	385
D.3 IPv6 multicast addresses	386
D.4 Optional values.....	386
D.5 Protocol addresses	386
Annex E (normative) Transport of PTP over IEEE 802.3 transports	387
E.1 General.....	387
E.2 Ethertype	387
E.3 Multicast media access control (MAC) addresses.....	387

E.4 majorSdoId field values.....	388
E.5 Optional values.....	388
E.6 Protocol addresses.....	388
Annex F (normative) Transport of PTP over DeviceNET.....	389
F.1 Protocol.....	389
F.2 message timestamp point.....	389
F.3 clockIdentity.....	389
F.4 PTP message formats.....	389
F.5 DeviceNet addressing for PTP.....	390
Annex G (normative) Transport of PTP over ControlNET.....	391
G.1 Protocol.....	391
G.2 clockIdentity.....	391
G.3 PTP message formats.....	391
G.4 ControlNet addressing for PTP.....	391
Annex H (normative) Transport of PTP over IEC 61158 Type 10.....	392
H.1 Background.....	392
H.2 Message specification.....	393
H.3 DLPDU of the IEC 61158 TYPE10.....	394
H.4 Encoding specifications.....	395
Annex I (normative) Default PTP Profiles.....	398
I.1 General.....	398
I.2 General requirements.....	398
I.3 Delay Request-Response Default PTP Profile.....	398
I.4 Peer-to-Peer Default PTP Profile.....	400
I.5 High-Accuracy Delay Request-Response Default PTP Profile.....	401
Annex J (normative) Performance monitoring options (optional).....	407
J.1 General.....	407
J.2 Timestamp monitoring.....	407
J.3 Additional parameters.....	410
J.4 Record data types.....	412
J.5 Data sets for performance monitoring.....	414
Annex K (informative) Suppression of rogue Announce messages.....	419
K.1 Example—Star topology.....	420
K.2 Example—PTP Network with a single loop with an odd number of PTP Instances in the loop	422
K.3 Example—More complex single loop PTP Network.....	423
K.4 Example—Linear chain.....	423
Annex L (normative) Layer-1 based synchronization performance enhancement (optional).....	425
L.1 General.....	425
L.2 Basic terms.....	426
L.3 Link Reference Model.....	427
L.4 L1Sync port characteristics.....	429
L.5 L1Sync data sets.....	431
L.6 L1Sync message exchange.....	434
L.7 L1Sync port operation specification.....	437
L.8 Optional parameters (option within this option).....	439
L.9 Link verification using Signaling messages (informative).....	444
Annex M (informative) Sub-nanosecond synchronization using the High Accuracy Default PTP Profile.....	445
M.1 General.....	445
M.2 Frequency loopback.....	445
M.3 Timestamping precision.....	447
M.4 Timestamping accuracy.....	448

M.5 Medium and its asymmetry	449
M.6 Timing characteristics	450
Annex N (informative) Calibration procedures	452
N.1 General.....	452
N.2 Theoretical background.....	455
N.3 Assumptions and requirements.....	457
N.4 Calibration procedures.....	459
Annex O (informative) Example inter-domain interactions	466
O.1 General.....	466
O.2 Sourcing timing to multiple domains.....	466
O.3 Providing timing to users (sinks) from multiple independent domains.....	467
O.4 Transferring time from PTP domain A to PTP domain B	468
O.5 Example use for external configuration of port state	473
Annex P (informative) Security	474
P.1 Overview, assumptions, and approach	474
P.2 Multipronged approach—detailed definition	474
Annex Q (informative) Bibliography	495
Annex R (informative) IEEE List of participants	498

IECNORM.COM : Click to view the full PDF of IEC 61588:2021

PRECISION CLOCK SYNCHRONIZATION PROTOCOL FOR NETWORKED MEASUREMENT AND CONTROL SYSTEMS

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC document(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. *IEEE documents are made available for use subject to important notices and legal disclaimers (see <http://standards.ieee.org/ipr/disclaimers.html> for more information).*

IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

- 2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE SA) Standards Board.
- 3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.
- 8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material 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. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

IEC 61588/IEEE Std 1588 was processed through IEC technical committee 65C: Industrial networks, under the IEC/IEEE Dual Logo Agreement. It is an International Standard.

The text of this International Standard is based on the following documents:

IEEE Std	FDIS	Report on voting
1588 (2019)	65C/1084/FDIS	65C/1091/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IECNORM.COM : Click to view the full PDF of IEC 61588:2021

Abstract: In this standard, a protocol is defined that provides precise synchronization of clocks in packet-based networked systems. Synchronization of clocks can be achieved in heterogeneous systems that include clocks of different inherent precision, resolution, and stability. The protocol supports synchronization accuracy and precision in the sub-microsecond range with minimal network and local computing resources. Customization is supported by means of profiles. The protocol includes default profiles that permit simple systems to be installed and operated without the need for user management. Sub-nanosecond time transfer accuracy can be achieved in a properly designed network.

Keywords: Boundary Clock, clock, Grandmaster Clock, IEEE 1588™, management, Ordinary Clock, security, synchronization, Transparent Clock

IECNORM.COM : Click to view the full PDF of IEC 61588:2021

Important Notices and Disclaimers Concerning IEEE Standards Documents

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

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

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

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

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

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

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

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

Translations

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

Official statements

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

Comments on standards

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

Comments on standards should be submitted to the following address:

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

Laws and regulations

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

Copyrights

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

Photocopies

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

Updating of IEEE Standards documents

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

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

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

Errata

Errata, if any, for IEEE standards can be accessed via <https://standards.ieee.org/standard/index.html>. Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore: <https://ieeexplore.ieee.org/browse/standards/collection/ieee/>. Users are encouraged to periodically check for errata.

Patents

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

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

Introduction

This introduction is not part of IEEE Std 1588-2019, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

This standard defines a protocol that provides precise synchronization of clocks in packet-based networked systems. The Precision Time Protocol (PTP) generates a master–slave relationship among the PTP Instances in the system. The clocks in all PTP Instances ultimately derive their time from a clock known as the “Grandmaster Clock.” In its basic form, this protocol is intended to be administration free.

IEEE Std 1588-2019 includes content that was not present in IEEE Std 1588-2008. Similarly some content that was present in the IEEE Std 1588-2008 is not in IEEE Std 1588-2019. The following Annexes in 1588-2008 are not present in IEEE Std 1588-2019:

- Annex C (informative) Examples of residence and asymmetry corrections
- Annex K (informative) Security protocol (experimental)
- Annex L (informative) Transport of cumulative frequency scale factor offset (experimental)

Acknowledgments

The working group would like to acknowledge Dieter Sibold and Steffen Fries for their contributions to the integrated security mechanism.

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

1. Scope

This standard defines a network protocol, the Precision Time Protocol (PTP), enabling accurate and precise synchronization of the real-time clocks of devices in networked distributed systems. The protocol is applicable to systems where devices communicate via networks, including Ethernet. The standard allows multicast communication, unicast communication or both. The standard specifies requirements for mapping the protocol to specific network implementations and defines such mappings, including User Datagram Protocol (UDP)/Internet Protocol (IP versions 4 and 6), and layer-2 IEEE 802.3 Ethernet.

The protocol enables heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize to a grandmaster clock. The protocol supports synchronization in the sub-microsecond range with minimal network bandwidth and local clock computing resources. The protocol enhances support for synchronization to better than 1 nanosecond. The protocol specifies how corrections for path asymmetry are made, if the asymmetry values are known. The grandmaster can be synchronized to a source of time external to the system, if time traceable to international standards or other source of time is required. The protocol provides information for devices to compute Coordinated Universal Time (UTC) from the protocol distributed time, if the grandmaster is traceable to international standards and is able to access pending leap-second changes. Options are also provided to allow end devices to compute other time scales from the protocol distributed time scale.

The protocol defines timing domains in which system timing is consistent. The protocol establishes the timing topology. The default behavior of the protocol allows simple systems to be installed and operated without requiring the administrative attention of users to determine the system timing topology.

The standard defines all needed data types, message formats, required computations, internal states, the behavior of devices with respect to transmitting, receiving, and processing protocol communications. The standard provides for the management of protocol artifacts in devices. The standard defines formal mechanisms for message extensions and the requirements for profiles that allow customization for specific application domains.

The standard defines conformance requirements. Optional specifications are provided for protocol security. This standard documents conditions under which this standard is backward compatible with IEEE 1588-2008.

2. Normative references

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

FIPS PUB 180-4:2015, Secure Hash Standard.¹

FIPS PUB 198-1:2008, The Keyed-Hash Message Authentication Code (HMAC).

IEC 61158-5-10:2007, Industrial communication networks—Fieldbus specifications—Part 5-10: Application layer service definition—Type 10 elements.²

IEC 61158-6-10:2007, Industrial communication networks—Fieldbus specifications—Part 6-10: Application layer protocol specification—Type 10 elements.

IEC 61784-1:2007, Industrial communication networks—Profiles—Part 1: Fieldbus profiles.

IEC 61784-2:2007, Industrial communications networks—Profiles—Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3.

IEC 62026-3:2008, Low-voltage switchgear and controlgear—Controller-device interfaces (CDIs)—Part 3: DeviceNet.

IEEE Std 754™, IEEE Standard for Binary Floating-Point Arithmetic.^{3, 4}

IEEE Std 802®, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture.

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

IEEE Std. 802c™, Standard for Local and Metropolitan Area Networks—Overview and Architecture Amendment: Local Medium Access Control (MAC) Address Usage.

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

IEEE Std 802.3™, IEEE Standard for Ethernet.

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

IERS Bulletin C.⁵

IETF RFC 768 (1980), User Datagram Protocol, Postel, J., ed.⁶

IETF RFC 791 (1981), Internet Protocol, Postel, J., ed.

¹ FIPS publications are available from the National Technical Information Service, U. S. Department of Commerce (<http://www.ntis.org/>).

² IEC publications are available from the International Electrotechnical Commission (<http://www.iec.ch>) and the American National Standards Institute (<http://www.ansi.org/>).

³ IEEE publications are available from The Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

⁴ The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

⁵ IERS publications are available from the International Earth Rotation and Reference Systems Services (<https://www.iers.org/>).

⁶ IETF publications are available from the Internet Engineering Task Force (<https://www.ietf.org/>).

IETF RFC 1624 (1994), Computation of the Internet Checksum via Incremental Update, Rijssinghani, A., ed.

IETF RFC 4868 (2007), Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec, Kelly, S., and Frankel, S., eds.

IETF RFC 8200 (2017), Internet Protocol, Version 6 (IPv6) Specification, Deering, S., and Hinden, R., eds.

ISO/IEC 10646:2003, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane.⁷

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply.

3.1.1 accuracy: The mean over an ensemble of measurements of the time or frequency error between the clock under test and a reference clock.

NOTE 1—Line “B” in Figure 1 represents the error in the measured mean value with respect to the reference, that is, the accuracy. The width of the curve of ensemble measurements, “C” in Figure 1, represents the measurement precision.⁸

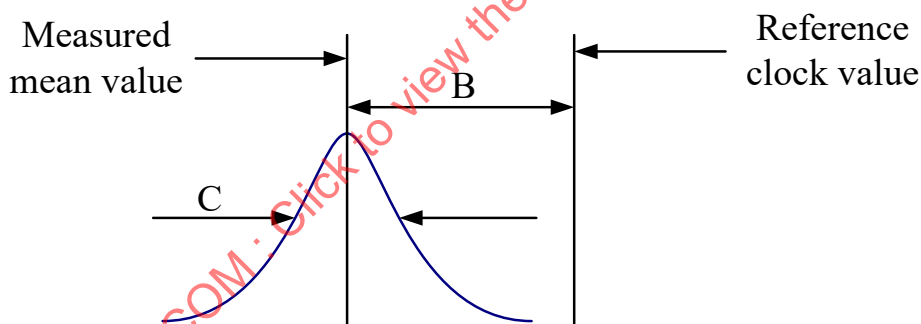


Figure 1—Histogram of an ensemble of measurements

NOTE 2—Different industries define "accuracy" in differing ways; however, for the discussion of an IEEE 1588 Grandmaster Clock, a smaller mean or maximum error from the reference time scale will be considered more accurate (whether the mean error or the maximum error is more relevant depends on the application).

3.1.2 atomic process: A process in which all the values of all inputs to the process are not permitted to change until all the results of the process are instantiated, and the outputs of the process are not visible to other processes until the processing of each output is complete.

⁷ISO publications are available from the International Organization for Standardization (<http://www.iso.org/>) and the American National Standards Institute (<http://www.ansi.org/>).

⁸Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

3.1.3 Boundary Clock: A PTP Instance that has multiple PTP Ports in a domain and maintains the timescale used in the domain. Within a domain, it may serve as the source of time to other PTP Instances, that is, be a Master Clock, and it can in addition synchronize to another Boundary Clock or Ordinary Clock, that is, be a Slave Clock.

3.1.4 clock: A device that can provide a measurement of the passage of time since a defined epoch.

NOTE 1—There are only two types of clocks specified in this standard: the Local PTP Clock and the Local Clock.

NOTE 2—For Local PTP Clocks associated with Ordinary Clocks and Boundary Clocks that are properly synchronized, the epoch is the epoch of the timescale in use. In the case of a Local Clock associated with a Transparent Clock, the epoch is locally defined and not necessarily aligned with the timescale in use.

NOTE 3—A clock provides time at desired moments of the timescale it maintains. Time is obtained either:

- *Physically:* in this type of clock, the time is modeled using a clock signal and a time counter that is driven by the clock signal;
- *Mathematically:* in this type of clock, the time is generated by a model that describes the relation of this clock to another clock (e.g., to a physical clock in a different timescale). The model enables the calculation of the time of the clock from the time of the other clock.

NOTE 4—The word “clock” appears many times in this standard modified as Boundary Clock, Ordinary Clock, Transparent Clock, or Grandmaster Clock. When used this way the combination, e.g. Boundary Clock, is not a clock as defined here but rather contains a clock. For historical reasons, these terms have been retained.

3.1.5 clock signal: A physical signal that has periodic events. The periodic events mark the significant instants at which a time counter is incremented. The clock signal is characterized by its frequency and phase.

3.1.6 default: When applied to attribute values and options of a PTP Instance, “default” means the configuration of a PTP Instance as it is delivered from the manufacturer.

3.1.7 device: An entity implementing some functionality, for example, a clock, a Boundary Clock, an encoder, and a port. *See also:* **PTP Instance**.

3.1.8 Direct PTP Link: A direct physical link with no intervening network elements between two PTP Ports of Boundary Clocks, Ordinary Clocks, and/or peer-to-peer Transparent Clocks.

NOTE 1—A direct physical link, where at least one end point is a PTP Port of an end-to-end Transparent Clock, is not a Direct PTP Link.

NOTE 2—A Direct PTP Link is a PTP Communication Path if the delay request-response mechanism is used. It is a PTP Link if the peer-to-peer delay mechanism is used, and is at the same time a PTP Communication Path if it is between two Boundary Clocks, two Ordinary Clocks, or a Boundary Clock and an Ordinary Clock.

3.1.9 domain (or PTP Domain): A logical grouping of PTP Instances using PTP to ensure that all Local PTP Clocks in the grouping are synchronized to the Grandmaster Clock of the domain, but are not necessarily synchronized to the Local PTP Clocks in another domain.

3.1.10 end-to-end port: A PTP Port that is configured to use the delay request-response mechanism.

3.1.11 end-to-end Transparent Clock: A Transparent Clock that supports the use of the delay request-response mechanism between a PTP Port in the MASTER state and a PTP Port in the SLAVE state in the same domain.

3.1.12 epoch: The origin of a timescale.

3.1.13 event: An abstraction of the mechanism by which signals or conditions are generated and represented. In this abstraction, the aspects of interest of the signals are conditions that occur at discrete instants of time.

3.1.14 foreign master: If an Announce message is received on PTP Port a of PTP Instance A from PTP Port b of PTP Instance B in the same PTP Network, and PTP Port b is not the current Master of PTP Port a, then PTP Port b is considered a foreign master by PTP Port a.

3.1.15 fractional frequency offset: The fractional frequency offset (FFO) between a measured frequency and a reference frequency is defined as follows:

$$\text{FFO} = \frac{(\text{FO} - \text{FR})}{\text{FR}} \quad (1)$$

where

FO is the measured frequency
FR is the reference frequency

3.1.16 frequency offset: The frequency offset between a measured frequency and a reference frequency is defined as follows:

$$\text{Frequency Offset} = \text{FO} - \text{FR} \quad (2)$$

where

FO is the measured frequency,
FR is the reference frequency.

3.1.17 Grandmaster Clock: In the context of a single PTP domain, the Local PTP Clock of an Ordinary Clock or a Boundary Clock that is the source of time to which all other Local PTP Clocks in the domain are synchronized.

3.1.18 Grandmaster PTP Instance: A PTP Instance containing the Grandmaster Clock.

3.1.19 holdover or holdover mode: A clock A, previously synchronized/syn-tonized to another clock B (normally a primary reference or a Master Clock) but whose frequency is determined in part using data acquired while it was synchronized/syn-tonized to B, is said to be in holdover or in the holdover mode as long as it is within its accuracy requirements.

3.1.20 implemented message timestamp point: A point within a Precision Time Protocol event message serving as a timestamp-capture point in the message. A timestamp is captured at the instant an implemented message timestamp point passes a point possibly removed from the reference plane. Depending on the implementation, the implemented message timestamp point is or is not equivalent to the message timestamp point. See: **message timestamp point**.

NOTE—The relationship between the two is specified in 7.3.4.2.

3.1.21 integrity check value (ICV): The result of a cryptographic function used to ensure that unauthorized modifications of a message are detected (e.g., the value might be the result of the keyed hash function HMAC-SHA256).

NOTE—See FIPS PUB 180-4:2015.

3.1.22 key: In the context of a security association, refers to the secret used in the cryptographic calculation of the integrity check value (ICV).

3.1.23 key distribution center (KDC): The functionality of a (typically) centralized authority to which peers authenticate before distribution of security parameters. For instance, KDCs are used to distribute security parameters for group-based communication.

3.1.24 L1 rx clock signal: A clock signal that is recovered from the reception of data from the medium.

3.1.25 L1 tx clock signal: A clock signal that is used in the transmission of data over the medium.

3.1.26 Local Clock: A physical clock that is used in the operation of the PTP Protocol.

- The Local PTP Clock may be derived from or identical to the Local Clock;
- The Local Clock may be syntonized to a reference clock signal.

NOTE—See 12.2.

3.1.27 Local Clock signal: The clock signal of a Local Clock.

3.1.28 Local PTP Clock: The clock of a PTP Instance that provides the local estimate of the time of its Grandmaster Clock, that is it is synchronized to the time of the Grandmaster Clock. It is either a physical or a mathematical clock, and it provides PTP or ARB (arbitrary) time.

NOTE—See 7.2.1.

3.1.29 Local PTP Clock signal: The clock signal of a Local PTP Clock.

3.1.30 management mechanism: A management protocol used for configuring and/or monitoring PTP Nodes and PTP Instances.

NOTE—This term refers to the PTP management messages of this standard, as well as to the management protocols defined in other standards (e.g., SNMP [B27]⁹ and NETCONF [B20]).

3.1.31 Master Clock: In the context of a single PTP Communication Path, the Local PTP Clock of an Ordinary Clock or Boundary Clock that is the source of time to which all other Local PTP Clocks on that PTP Communication Path synchronize.

3.1.32 Master PTP Instance: A PTP Instance containing a Master Clock.

NOTE—A Master PTP Instance propagates the traceability information from the Grandmaster PTP Instance throughout the associated PTP Communication Path.

3.1.33 message timestamp point: A point within a PTP event message serving as a reference point in the message. A timestamp is defined by the instant a message timestamp point passes the reference plane of a PTP Instance.

3.1.34 mixed-step PTP Instance: A PTP Instance where some PTP Ports can be one-step PTP Ports and one or more PTP Ports can be two-step PTP Ports, see one-step PTP Port and two-step PTP Port.

3.1.35 multicast communication: A communication model in which each PTP message sent from any PTP Port is received and processed by all PTP Ports on the same PTP Communication Path.

⁹ The numbers in brackets correspond to those of the bibliography in Annex Q.

3.1.36 Mutable field: A field that can be modified on retransmission by PTP Instances. Handling of mutable fields in the context of security depends on the key management system.

3.1.37 one-step port semantics: Semantics where a transmitted Sync or Pdelay_Resp message will not be followed by a Follow_Up or Pdelay_Resp_Follow_Up message, respectively.

3.1.38 one-step PTP Instance: A PTP Instance where all PTP Ports are one-step PTP Ports.

3.1.39 one-step PTP Port: A PTP Port that obeys one-step port semantics.

3.1.40 Ordinary Clock: A PTP Instance that has a single PTP Port in its domain and maintains the timescale used in the domain. An Ordinary Clock can serve as a source of time, that is, contain a Master Clock, or alternatively, the Local PTP Clock of an Ordinary Clock can be synchronized, that is, be a Slave Clock, to the Local PTP Clock of a Boundary Clock or another Ordinary Clock in the domain.

3.1.41 Parent PTP Instance: Within a domain, an Ordinary Clock or Boundary Clock A, is the Parent PTP Instance of another Ordinary Clock or Boundary Clock B, if the Local PTP Clock of A is the Master Clock of the Slave Clock in B.

3.1.42 peer-to-peer port: A PTP Port that is configured to use the peer-to-peer delay mechanism.

3.1.43 peer-to-peer Transparent Clock: A Transparent Clock that, in addition to providing PTP event transit time information, also corrects for the propagation delay of the PTP Link connected to the PTP Port receiving the Sync message. In the presence of peer-to-peer Transparent Clocks, delay measurements between Slave Clocks and the Master Clock are performed using the peer-to-peer delay mechanism.

3.1.44 phase change rate: The observed rate of change in the measured time with respect to the reference time. The phase change rate is equal to the frequency offset between the measured frequency and the reference frequency. *See also: frequency offset*

3.1.45 phase offset: Let $T_A(t)$ and $T_B(t)$ be the times indicated by clocks associated with a clock signal A and a clock signal B, respectively, as a function of time t . Then the phase offset of A with respect to B is $T_A(t) - T_B(t)$. A positive value of the phase offset means that A is in advance of B. Phase offset can exist between clocks, clock signals, as well as a clock and a clock signal.

NOTE—Evaluation of the phase offset does not require an existence of clocks associated with the respective clock signals. A phase offset can be directly evaluated by measuring the time interval between the relevant periodic events of clock signals A and B. Given the definition, in such a measurement, a positive value of the phase offset means that the periodic event of the clock signal A occurs earlier in time than the corresponding event of the clock signal B.

3.1.46 portNumber: An index identifying a specific PTP Port on a PTP Instance.

3.1.47 precision: See: **accuracy**.

3.1.48 Precision Time Protocol (PTP): The protocol defined by IEEE Std 1588™. As an adjective, it indicates that the modified noun is specified in or interpreted in the context of IEEE Std 1588™.

3.1.49 primary reference: A source of time and/or frequency that is traceable to international standard(s). *See also: traceability*.

3.1.50 Profile (or PTP Profile): A document, or a portion of a document, specifying the set of PTP features and attribute values applicable to a PTP Instance, and written by an organization following the specifications of 20.3 of IEEE Std 1588™-2019.

3.1.51 PTP Communication: Information used in the operation of the PTP protocol, transmitted in a PTP message.

3.1.52 PTP Communication Path: Within a domain, the signaling path portion of a particular PTP Network connecting Ordinary Clocks and Boundary Clocks with no intervening Boundary Clocks. However, there can be Transparent Clocks or non-PTP devices in a PTP Communication Path.

NOTE—See Figure 2.

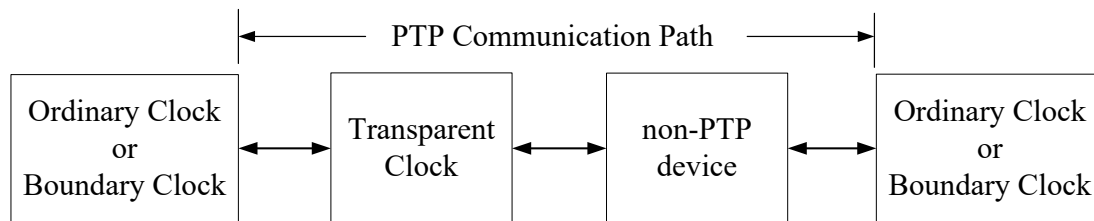


Figure 2—PTP Communication Path

3.1.53 PTP Instance: An instance of the PTP protocol, operating in a single device, within exactly one domain. A PTP Instance implements those portions of the standard indicated as applicable to an Ordinary Clock, a Boundary Clock, or a Transparent Clock.

3.1.54 PTP Instance Time: Time measured on the timescale maintained by the Local PTP Clock of the PTP Instance. The timescale of a Local PTP Clock is either the timescale PTP or the timescale ARB. See 7.2.1 of IEEE Std 1588™-2019.

NOTE—PTP Instance Time is not available in a Transparent Clock that does not maintain a Local PTP Clock.

3.1.55 PTP Link: Within a domain, a network segment between two PTP Ports using the peer-to-peer delay mechanism of this standard. The peer-to-peer delay mechanism is designed to measure the propagation time over such a link.

NOTE 1—In many applications, this network segment can be a direct physical link between the PTP Nodes including the above PTP Ports.

NOTE 2—A PTP Link between PTP Ports of Boundary Clocks or Ordinary Clocks is also a PTP Communication Path, see PTP Communication Path.

3.1.56 PTP management message: A PTP message defined in IEEE Std 1588™ for the purpose of configuring and/or monitoring PTP Nodes and PTP Instances.

NOTE—This term does not refer to messages of management protocols defined in other standards (e.g., SNMP and NETCONF).

3.1.57 PTP Management Node: A device that configures and/or monitors Boundary Clocks, Ordinary Clocks and/or Transparent Clocks in one or more domains using the PTP management messages of IEEE Std 1588.

3.1.58 PTP message: One of the messages defined in IEEE Std 1588™.

NOTE—See 6.4.

3.1.59 PTP Network: A network consisting of a combination of PTP Nodes and possibly non-PTP devices and/or PTP Management Node(s). Non-PTP devices include, for example, some bridges, routers, and other infrastructure devices, and possibly devices such as computers, printers, and other application devices.

3.1.60 PTP Node: A device that contains one or more PTP Instances and/or PTP services (e.g., Common Mean Link Delay Service).

3.1.61 PTP Port: A logical access point of a PTP Instance for PTP Communication to the communications network.

3.1.62 recognized standard time source: A source external to Precision Time Protocol (PTP) that provides time and/or frequency as appropriate, and that is traceable to the international standards laboratories maintaining clocks that form the basis for the International Atomic Time (TAI) and Coordinated Universal Time (UTC) timescales. Examples of these are GNSS systems (e.g. GPS, GLONASS), and national time laboratories (e.g. National Institute of Standards and Technology in the US, National Physical Laboratory in the UK).

3.1.63 requester: In the context of the peer-to-peer delay mechanism, the PTP Port implementing the peer-to-peer delay mechanism that initiates the operation of the mechanism by sending a Pdelay_Req message.

3.1.64 responder: In the context of the peer-to-peer delay mechanism, the PTP Port responding to the receipt of a Pdelay_Req message as part of the operation of the peer-to-peer delay mechanism.

3.1.65 retransmit: In the context of a PTP Instance processing PTP messages (e.g. the PTP stack running above the MAC layer), retransmission of a PTP message means that the PTP-defined contents of a received PTP message, possibly updated, is the PTP-defined content of new PTP message(s) transmitted on one or more PTP Ports distinct from the PTP Port on which the original PTP message was received.

3.1.66 Security Association (SA): Description of the set of security parameters necessary to provide security services (e.g., authentication and integrity) between different entities sharing the same SA. The security parameters of the SA are typically negotiated or signaled in the context of the key management protocol.

3.1.67 Security Association Database (SAD): Database containing the sets of SAs with their associated security parameters. The content of the database is dynamic in the context of the application being secured.

3.1.68 Security Parameter Pointer (SPP): Identifier of a specific SA. If carried in the application protocols it is used to point to the associated SA in the SAD and the SPD.

3.1.69 Security Policy Database (SPD): Database containing the required security services (e.g., message integrity required for all messages) with their specifics (e.g., at least support for HMAC-SHA256 and an associated key for ICV calculation) for an application. The SPD is typically involved in the establishment of an SA to ensure the adherence to a given policy. The SPD is configured and content is static between any reconfiguration.

3.1.70 Special Port: A Special Port is a PTP Port that implements a mechanism for transferring time over a network where the time transfer is not based on the use of PTP timing messages.

NOTE—See 11.5.1.

3.1.71 Slave Clock: In the context of a single PTP Communication Path, the Local PTP Clock of an Ordinary Clock or Boundary Clock that synchronizes to the Local PTP Clock of the Master PTP Instance on that PTP Communication Path.

3.1.72 Slave PTP Instance: A PTP Instance containing a Slave Clock.

NOTE—A Boundary Clock can be both a Master PTP Instance and a Slave PTP Instance. See **Master PTP Instance**, **Slave PTP Instance** and **Boundary Clock**.

3.1.73 specified-by-design: An attribute, field, or state variable with a value inherent in the implementation or specified by IEEE Std 1588™ or the applicable PTP Profile as opposed to the case where the value is subject to change by configuration or the operation of the protocol.

3.1.74 stability (of a clock or clock signal): A measure of the variations over time of the frequency error (of the clock or clock signal). The frequency error typically varies with time due to aging and various environmental effects, for example, temperature.

3.1.75 synchronized clocks: Absent relativistic effects, two clocks are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of the same single event occurring at an arbitrary instant differ by no more than that uncertainty.

3.1.76 syntonized clocks: Absent relativistic effects, two clocks are syntonized to a specified uncertainty if the duration of the second is the same on both, which means the time as measured by each advances at the same rate within the specified uncertainty. The two clocks might or might not share the same epoch.

3.1.77 time counter: A digital time representation of the time that is incremented by the period of a clock signal at each periodic event of the clock signal. *See:* **clock signal**.

3.1.78 timeout: A mechanism for terminating requested activity that, at least from the perspective of the entity requesting termination, does not complete within the specified time.

3.1.79 timescale: A measure of elapsed time since an epoch.

3.1.80 Timestamping Clock: The Timestamping Clock is used in generating ingress and egress timestamps for PTP event messages and is either the Local PTP Clock or the Local Clock.

NOTE—See: 7.3.4.3.

3.1.81 traceability: A property of the result of a measurement or the value of a standard whereby it can be related to stated references, usually national or international standards, through an unbroken chain of comparisons all having stated uncertainties. (Adapted from the International Vocabulary of Basic and General Terms in Metrology Second Edition 1993 Definition 6.10). For purposes of this standard, the term “unbroken chain of comparisons” means that if measurements were performed comparing the difference between the stated reference and any point in the chain (for example, the time difference between time in the GNSS system and the time in a Grandmaster Clock), the results of these measurements would be within the stated uncertainties. For example, as long as a measurement, if performed, were known to be within the stated uncertainty, it does not matter whether the Grandmaster PTP Instance is actively disciplining its Local PTP Clock to the GNSS system or whether the Local PTP Clock is in holdover.

3.1.82 traceable: The time or frequency of a clock if measurements of these quantities have the traceability property.

3.1.83 translation device: A Boundary Clock or, in some cases, a Transparent Clock that translates the PTP messages between regions implementing different transport mappings, between different versions of this standard, or different Precision Time Protocol (PTP) Profiles.

3.1.84 Transparent Clock: A PTP Instance that measures the time for a PTP event message to transit the PTP Instance and provides this information to PTP Instances receiving this PTP event message. Peer-to-peer Transparent Clocks also correct for PTP Link delay. *See also:* **end-to-end Transparent Clock** **peer-to-peer Transparent Clock**

3.1.85 two-step port semantics: Semantics where a transmitted Sync or Pdelay_Resp message will be followed by a Follow_Up or Pdelay_Resp_Follow_Up message, respectively.

3.1.86 two-step PTP Instance: A PTP Instance where all PTP Ports are two-step PTP Ports.

3.1.87 two-step PTP Port: A PTP Port that obeys two-step port semantics.

3.2 Acronyms and abbreviations

ARB	arbitrary
BMCA	best master clock algorithm
CID	Company identification (allocated by the IEEE; see 7.5.2.2.1)
CMLDS	Common Mean Link Delay Service
CP	Communication Profile [according to IEC 61784-1:2007]
CPF	Communication Profile Family [according to IEC 61784-1:2007]
E2E	end-to-end
EEC	synchronous Ethernet Equipment Clock
EEC1	EEC Option 1
EEC2	EEC Option 2
EPON	Ethernet Passive Optical Network
GDOI	Group Domain of Interpretation
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IANA	Internet Assigned Numbers Authority
ID	identification
IEC	International Electrotechnical Commission
IEEE RA	IEEE Registration Authority
IERS	International Earth Rotation and Reference Systems Service
IETF	Internet Engineering Task Force
IPv4, IPv6	Internet Protocol version 4 / 6
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
L1	layer-1
MAC	media access control [according to IEEE Std 802.3]
MDMI	Media-Dependent, Media-Independent
MJD	Modified Julian Day
MTIE	Maximum Time Interval Error
MUC	Medium Under Calibration
NIST	National Institute of Standards and Technology (see www.nist.gov)
NTP	Network Time Protocol (see IETF RFC 5905 [B19])
NUI	Network Unique Identifier
OID	Object Identifier
OUI	Organizationally Unique Identifier (allocated by the IEEE; see 7.5.2.2.1)
P2P	peer-to-peer
PDU	Protocol Data Unit
PHY	physical layer [according to IEEE Std 802.3]
PIUC	PTP Instance Under Calibration
PLL	Phase Locked Loop
POSIX	Portable Operating System Interface (see ISO/IEC 9945 [B30])
ppm	parts per million
PPS	pulse per second
PTP	Precision Time Protocol
QSDO	Qualified Standards Development Organization
RFC	Request For Comments
rx	receive
SFP	small form-factor pluggable
TAI	International Atomic Time
TC	traffic class
TDEV	Time deviation
TESLA	Time Efficient Stream Loss-Tolerant Authentication
TLV	type, length, value
tx	transmit
UCMM	UnConnect Message Manager

UDP/IP	User Datagram Protocol [see IETF RFC 768 (1980)]/Internet Protocol [see IETF RFC 791 (1981)]
UTC	Coordinated Universal Time

IECNORM.COM : Click to view the full PDF of IEC 61588:2021

4. Conventions

4.1 Descriptive lexical form syntax

4.1.1 Lexical form syntax

A lexical form refers to:

- a) A name,
- b) A data type.

The conventions illustrated in the following list regarding lexical forms are used in this standard:

- **Type names:** For example, ClockQuality (no word separation, initial letter of each word capitalized).
- **Enumeration members and global constants:** For example, ATOMIC_CLOCK (underscore word separation, all letters capitalized).
- **Fields within PTP messages, instances of structures, and variables:** For example, secondsField, clockQuality, clockIdentity (two-word field names at a minimum, no word separation, initial word not capitalized, initial letter capitalization on subsequent words).
- **Members of a structure:** For example, clockQuality.clockClass (structure name followed by a period followed by the member name).
- **Data set name:** For example, defaultDS, parentDS, portDS, currentDS, timePropertiesDS (no word separation, initial letter of each word not capitalized followed by the letters DS).
- **Data set members:** For example, defaultDS.clockQuality.clockClass (data set name followed by a period followed by a member name followed by a period followed by a member name) or defaultDS.instanceEnable (data set name followed by a period followed by a member name).
- **PTP message names:** For example, Sync, Delay_Req (underscore word separation, initial letter of each word capitalized).
- **<localNameForSomething>:** Text enclosed in angle brackets < > is used where the standard needs to refer to something whose syntax or lexical form is dependent on the local implementation and language, for example, in 5.3.1 in defining data types. It is also used to express an abstract quantity as part of a description, for example, <delayAsymmetry> in 7.4.2 in connection with Figure 27, or as a computed quantity used in the operation of the protocol, for example, <delayAsymmetry> in 8.2.15.4.8 in defining the value of portDS.delayAsymmetry, or in 10.2.2.1.1 for computing the value of the correctionField of a Sync message. The <localNameForSomething> is also used when discussing artifacts, for example, measurements or computations, in the abstract as opposed to normative specifications. Angle bracket notation is also used in defining a type, to indicate what is filled in. For example: struct <StructureName> (see 5.3.1).
- **Names of entities and subsystems in the model architecture:** For example, Boundary Clock, PTP Node, Clock Source, PTP Common Core (separate words all with initial Capitalization).

When a lexical form appears in text, as opposed to in a type, or a format definition, the form is to be interpreted as singular, plural, or possessive as appropriate to the context of the text.

4.2 Word usage

4.2.1 Shall

The word “shall,” which is equivalent to “is required to,” is used to indicate mandatory requirements, strictly to be followed to conform to the standard and from which no deviation is permitted.

4.2.2 Must

The word “must” indicates an unavoidable situation.

4.2.3 Should

The word “should,” which is equivalent to “is recommended that,” is used to indicate the following:

- Among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others.
- That a certain course of action is preferred but not required.
- That (in the negative form) a certain course of action is deprecated but not prohibited.

4.2.4 May

The word “may,” which is equivalent to “is permitted,” is used to indicate a course of action permissible within the limits of the standard.

4.2.5 Can

The word “can,” which is equivalent to “is able to,” is used to indicate possibility and capability, whether material or physical.

4.2.6 Optional

Clauses, annexes, and text marked optional are not required to be implemented. If the option is implemented, then all specifications of the clause, annex, or text, respectively, shall be implemented according to this standard.

NOTE—This definition is recursive, which means that options within options obey these same rules.

4.2.7 Reserved

The word “reserved” indicates:

- If used in an assignment of values to an enumeration or an attribute that the values indicated are reserved for use in future editions of this standard and shall not be used for any other purpose.
- If used in a field of a PTP message that the field is reserved for use in future editions of this standard. The field shall be present in the PTP message with the size specified. No interpretation of reserved fields is to be made for this edition of this standard, and the fields shall not be used for any other purpose.

4.2.8 Deprecated

The term “deprecated” indicates that a feature of the standard is permitted to be used in compliant devices, but it is no longer recommended or required. The function realized by the deprecated feature can now be realized using an alternate, superior option. Deprecated features may serve an important role in maintaining backward compatibility.

4.2.9 Experimental

The word “experimental” indicates that:

- Any artifact designated in this standard as experimental is intended for use in conjunction with protocol development within a privately administered development network, for example, within a network that has no wide area connectivity. Within that network, a local administrator is free to use artifacts marked as experimental for protocol development purposes.
- These experimental artifacts shall not be used in protocols or products that are to be released for use in the wider networking community, as freeware, shareware, or any part of a company’s commercial product offering, and shall not be specified in a standard or be permanently assigned for use with a given protocol or protocols.
- Prior to deployment in an environment outside the developing organization’s administrative control or for public release as a standard or product, the experimental artifact and any specifications dependent on the artifact shall either:
 - Be submitted to and approved by the IEEE 1588 Working Group¹⁰ as a part of the IEEE 1588 standard. In this case, the specification of the artifact and any dependent specifications shall be a part of IEEE Std 1588 and not of any other standard, or
 - Be specified in a PTP Profile subject to the restrictions of 20.3.1.2, for example, as an alternate BMCA (see 9.3.1), or organization-specific TLV (see 14.3)
- Implementers must be aware that experimental artifacts, for example, experimental TLV values (see Table 52 in 14.1.1) or experimental sdoId values (see Table 2 in 7.1.4), can be used by anyone subject to the conditions of this clause. The use of such artifacts in violation of 4.2.9 can lead to interoperability issues in PTP Networks.

4.3 Behavioral specification notation

State transition diagrams are used to specify behavioral characteristics as illustrated in Figure 3. Each state transition diagram is composed of the following components:

- Named boxes, representing states
- Directed arrows, indicating transitions from one state to the next

Each transition is labeled with:

- a) Enabling event or predicate label for a transition
- b) Transition action label for a transition

¹⁰ IEEE Precise Networked Clock Synchronization Working Group. Also referred to in this document as the IEEE 1588 Working Group (<https://standards.ieee.org/project/1588.html>).

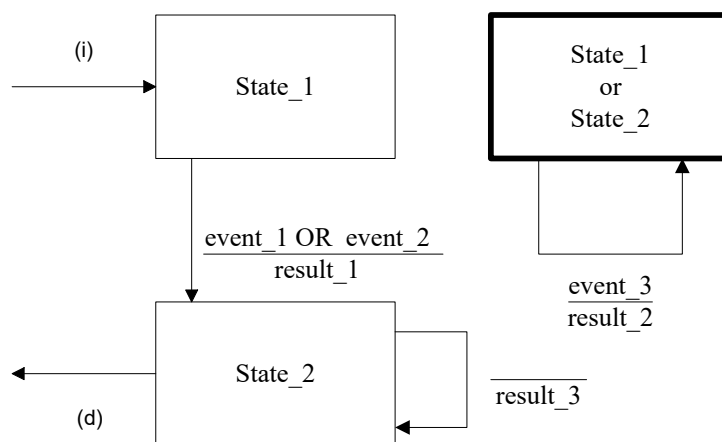


Figure 3—Mealy state transition diagram

The notation used describes state transition diagrams using the Mealy style, where actions are associated with the transition from one state into another.

Events, for example, “event_1,” “event_2,” and “event_3,” identify the inputs to the state machine. They can be operation requests and responses or internal occurrences such as timer expirations.

Predicates, for example, “event_1 OR event_2,” identify enabling conditions for transitions. The first predicate encountered, evaluated from left to right, which is TRUE, selects the transition to execute and therefore the next state.

Transition actions, for example, “result_1,” are the actions that are executed before transitioning to the next state.

The next state identifies the state for the state machine after the selected transition action completes. The value of the current state changes as the transition to the next state occurs.

A bold line for a state box indicates that the box represents multiple states. Any transition shown that begins and terminates in such a state box indicates that there has been no change in state.

Transitions, for example, the transition resulting in result_3, which have no indicated enabling conditions, occur via unspecified mechanisms. Unless otherwise stated in this standard, the events giving rise to these mechanisms are implementation specific and outside the scope of the standard.

A transition into a state machine, for example, “(i),” is indicated by a transition arrow that has no source state. A transition out of a state machine, for example, “(d),” is indicated by a transition arrow with no destination state.

NOTE—For example, as a result of either event_1 or event_2 becoming TRUE, State_1 is replaced with the value of the next state. In this example, the next state is State_2, which is specified as the name of the state box that is the target of the transition arrow. Before the transition, result_1 occurs. “event_3” can occur in either State_1 or State_2. The state is unchanged, but an action, result_2, occurs as the result of event_3.

5. Data types and on-the-wire formats

5.1 General

The data types specified for the various PTP variables and PTP message fields define logical properties that are necessary for correct operation of the protocol or interpretation of PTP message content.

Implementations are free to use any internal representation of PTP data types; however, the internal representation shall not change the semantics of any quantity visible via PTP Communications or the semantics of any specified operation of the protocol.

5.2 Primitive data type specifications

All nonprimitive PTP data types are derived from the primitive types listed in Table 1. These types are not tied to any specific programming language. The essential properties of each type shall be as follows:

- **Integer:** All integer types are of finite length as indicated by the number associated with each, for example, UInteger48, and as signed or unsigned as indicated by the absence or presence of a leading U. Numbers with these data types obey the laws of arithmetic within the range represented by the length. Arithmetic operations are treated as modulo the capacity of the data type; for example, the sum of two UInteger48 values is computed modulo 2^{+48} . Signed integers are represented in two's complement form.
- **Enumeration:** All enumerations are of finite field length as indicated by the number associated with each, for example, Enumeration4. Unless otherwise stated in this standard, the only interpretations of the bit pattern in the enumeration field are the associations between the bit patterns and the assigned meanings of the enumeration.
- **Boolean:** The only interpretation is as logical values within the context of Boolean algebra.
- **Nibble and Octet:** These are 4- and 8-bit fields, respectively. The only interpretations are those explicitly defined within this standard.
- **Float:** IEEE 754™ floating-point number, with a length that specifies the format. Support for floating-point data types is optional because usage of those data types is limited to the specifications of optional features.

Table 1—Primitive PTP data types

Data type	Definition
Boolean	TRUE or FALSE
Enumeration4	4-bit enumerated value
Enumeration8	8-bit enumerated value
Enumeration16	16-bit enumerated value
UInteger4	4-bit unsigned integer
Integer8	8-bit signed integer
UInteger8	8-bit unsigned integer
UInteger12	12-bit unsigned integer
Integer16	16-bit signed integer
UInteger16	16-bit unsigned integer
Integer32	32-bit signed integer
UInteger32	32-bit unsigned integer
UInteger48	48-bit unsigned integer
Integer64	64-bit signed integer
Nibble	4-bit field not interpreted as a number
Octet	8-bit field not interpreted as a number
Float64	IEEE 754 binary64 (64-bit double-precision floating-point format)

5.3 Derived data type specifications

5.3.1 General

Arrays of any of the primitive data types are represented in the format <data type>[lengthField] <label>, where [lengthField] indicates the number of instances of the data type in the array and <label> is the lexical name for the array data type so defined.

Structures consisting of an ordered list of members are indicated with the following syntax:

```
Struct <StructureName>
{
  <DataType1> <memberName1>;
  <DataType2> <memberName2>;
  ...
};
```

where <StructureName> is the lexical name for the data type so defined, <DataType1> is the data type of the first member, <memberName1> is the lexical name of the first member, and so forth.

The syntax Typedef <DataType> <TypeName> is interpreted as defining a derived data type with the same properties as the data type defined by <DataType> but with a new name given by <TypeName>.

The syntax Typedef <DataType> [lengthField] <TypeName> is interpreted as defining a derived data type consisting of an array of elements of type <DataType> but with a new name given by <TypeName>.

For the derived data types, the specifications of the semantics and values of all members of structs and arrays are found in the clauses specifying the use of the struct or array. This subclause specifies only the data types.

5.3.2 TimeInterval

The TimeInterval type represents time intervals.

```
typedef Integer64 TimeInterval;
```

TimeInterval is the time interval expressed in nanoseconds, multiplied by 2^{+16} .

Positive or negative time intervals outside the maximum range of this data type shall be encoded as the largest positive and negative values of the data type, respectively.

For example, 2.5 ns is expressed as 0000 0000 0002 8000₁₆.

NOTE—In the 2008 edition of this standard, TimeInterval was defined as a Struct. There are no backward compatibility issues because the on-the-wire format is the same.

5.3.3 Timestamp

The Timestamp type represents a positive time with respect to the epoch.

```
Struct Timestamp
{
    UInteger48 secondsField;
    UInteger32 nanosecondsField;
};
```

The secondsField member is the integer portion of the timestamp in units of seconds.

The nanosecondsField member is the fractional portion of the timestamp in units of nanoseconds.

The nanosecondsField member is always less than 10^9 .

For example:

+2.000000001 seconds is represented by secondsField = 0000 0000 0002₁₆ and nanosecondsField = 0000 0001₁₆.

5.3.4 ClockIdentity

The ClockIdentity type identifies unique entities within a PTP Network, for example, a PTP Instance or an entity of a common service.

```
typedef Octet[8] ClockIdentity;
```

5.3.5 PortIdentity

The PortIdentity type identifies a PTP Port (see 7.5) or a Link Port (see 16.6.1).

```
Struct PortIdentity
{
    ClockIdentity clockIdentity;
    UInteger16 portNumber;
};
```

5.3.6 PortAddress

The PortAddress type represents the protocol address of a PTP Port.

```
Struct PortAddress
{
  Enumeration16 networkProtocol;
  UInteger16 addressLength;
  Octet[addressLength] addressField;
};
```

The value of the networkProtocol member shall be taken from the networkProtocol enumeration (see 7.4.1).

The addressLength is the length in octets of the address. The range shall be 1 to 16 octets.

The addressField member holds the protocol address of a PTP Port in the format defined by the mapping annex of the protocol as identified by the networkProtocol member. The most significant octet of the addressField is mapped into the octet of the addressField member with index 0.

5.3.7 ClockQuality

The ClockQuality represents the quality of a clock.

```
Struct ClockQuality
{
  UInteger8 clockClass;
  Enumeration8 clockAccuracy;
  UInteger16 offsetScaledLogVariance;
};
```

5.3.8 TLV

The TLV type represents TLV extension fields.

```
Struct TLV
{
  Enumeration16 tlvType;
  UInteger16 lengthField;
  Octet [lengthField] valueField;
};
```

The length of all TLVs shall be an even number of octets.

5.3.9 PTPText

The PTPText data type is used to represent textual material in PTP messages.

```
Struct PTPText
{
  UInteger8 lengthField;
  Octet[lengthField] textField;
};
```

The textField member shall be encoded as UTF-8 symbols as specified by ISO/IEC 10646:2003. The most significant octet of the leading text symbol shall be the element of the array with index 0.

NOTE—A single UTF-8 symbol can be 1 to 4 octets long. Therefore, the lengthField value can be larger than the number of symbols. A symbol is analogous to a character.

PTPText is also used to represent textual strings in management mechanisms that do not use PTP messages. When used in this context, PTPText shall represent the string data type of the corresponding management standards (not the preceding struct).

5.3.10 FaultRecord

The FaultRecord type is used to construct fault logs.

```
Struct FaultRecord
{
    UInteger16 faultRecordLength;
    Timestamp faultTime;
    Enumeration8 severityCode;
    PTPText faultName;
    PTPText faultValue;
    PTPText faultDescription;
};
```

The faultRecordLength member shall indicate the number of octets in the FaultRecord not including the 2 octets of the faultRecordLength member.

5.3.11 RelativeDifference

The RelativeDifference type represents the relative difference between two numeric values.

```
Typedef Integer64 RelativeDifference;
```

RelativeDifference is the relative difference expressed as a dimensionless fraction and multiplied by 2^{+62} , with any remaining fractional part truncated.

NOTE 1—For example, 0.00024 is expressed as 0003 EEA2 09AA A3AD₁₆. The maximum range of RelativeDifference is (-2^{+1}) to $(2^{+1}-2^{-62})$, which is sufficient to store a fractional number.

NOTE 2—When using values with the data type RelativeDifference, in calculations in which other values are expressed in units of 2^{-16} ns (data type TimeInterval) or nanoseconds, the values need to be appropriately scaled before performing the calculations. In some implementations, it might be desired to avoid floating point arithmetic when working with expressions that involve the two data types, that is, RelativeDifference and TimeInterval. This can be accomplished by appropriately scaling the input raw values (expressed in dimensionless units of 2^{+62} , units of 2^{-16} ns, or units of nanoseconds) or the product of the computation. In the latter case, the raw values can be used if the result of the computation is appropriately scaled. For example, if a quantity of data type RelativeDifference is multiplied by a quantity with units of nanoseconds, and the result is desired in units of 2^{-16} ns, the raw values can be multiplied and then the product can be divided by 2^{+46} to obtain the result in units of 2^{-16} ns.

5.3.12 CommunicationCapabilities

The CommunicationCapabilities structure lists the communication modes in which a PTP Port can operate.

```
Struct CommunicationCapabilities
{
Boolean multicastCapable;
Boolean unicastCapable;
Boolean unicastNegotiationCapable;
Boolean unicastNegotiationRequired;
};
```

5.4 On-the-wire formats

5.4.1 General

PTP protocol data units consist of the PTP messages defined in this standard. The internal ordering of the fields of the PTP protocol data units is specified in 5.4.2 to 5.4.4.

5.4.2 Primitive data types

Numeric primitive data types defined in 5.2 shall be formatted with the most significant octet nearest the beginning of the protocol data unit followed in order by octets of decreasing significance.

The Boolean data type TRUE shall be formatted as a single bit equal to 1 and FALSE as a single bit equal to 0.

Enumerations of whatever length shall be formatted as though the assigned values are unsigned integers of the same length; for example, Enumeration16 shall be formatted as though the value had type UInteger16.

5.4.3 Arrays of primitive types

All arrays shall be formatted with the member having the lowest numerical index closest to the start of the protocol data unit followed by successively higher numbered members. In octet arrays, the octet with the lowest numerical index is termed the most significant octet.

When a field containing more than one octet is used to represent a numeric value, the most significant octet shall be nearest the start of the protocol data unit, followed by successively less significant octets.

When a single octet contains multiple fields of primitive data types, the bit positions within the octet of each of the primitive types as defined in the PTP message field specification shall be preserved. For example, the first field of the header of the PTP messages is a single octet composed of two fields, one of type Nibble bits 4 to 7, and one of type Enumeration4 bits 0 to 3 (see 13.3.1).

5.4.4 Derived data types

Derived data types defined as structs shall be formatted with the first member of the struct closest to the beginning of the protocol data unit followed by each succeeding member. Each member shall be formatted according to its data type.

Derived data types defined as typedefs shall be formatted according to its referenced data type.

5.4.5 Mapping of PTP protocol data units into their on-the-wire formats

Unless otherwise specified, PTP protocol data units shall be mapped to and from their on-the-wire format based on the rules of the underlying physical layer transport. Any exceptions are specified in one of the transport-specific annexes of this standard or in the applicable PTP Profile.

NOTE—PTP protocol mechanisms operate in the upper layers of the protocol stack (i.e., PTP is an “application” that uses the services of the network or link layer). The physical layer transport dictates the on-the-wire format.

6. Clock synchronization model

6.1 General requirements on implementations

6.1.1 General

This clause provides a model for understanding the operation of the Precision Time Protocol. The exact specifications for this operation are found in subsequent clauses.

This standard specifies a detailed model of the Precision Time Protocol. Implementers of PTP Instances may use internal representations, organizations, and procedures other than those presented in this document. The externally visible behavior of implementations of PTP Instances shall, in all respects, be identical to the behavior, including the externally visible results of computations, timing considerations, state evolutions, PTP message formats, PTP message contents, and data types, as specified by the models and specifications of this standard.

This standard provides the following classes of alternatives and options:

- Alternative transport mechanisms, of which at least one must be chosen (e.g., Annex C, Annex D, and Annex E)
- Alternative specifications for implementing normative requirements, for example, flexibility in conveying information in blade architectures, and the choice between methods of determining message propagation time
- Optional features that aid in constructing applications, for example, security or high accuracy
- Optional features that fine tune or enhance performance of the best master clock algorithm, for example, unicast discovery

In all cases, which of the alternatives is applicable, or which of the options is implemented and/or enabled/disabled, shall be specified by one or more of the following:

- The manufacturer
- The applicable PTP Profile
- One of the optional features specifying that another option or alternative be used

If an option or alternative is implemented, any associated optional data sets shall be implemented irrespective of whether the option or alternative is enabled or disabled. In this context, “implemented” means that a PTP Instance contains the executable, that is, the code, data sets, hardware, or whatever is required to execute the option or alternative. In the absence of an enable/disable feature and if the option is implemented, the executable is present and functioning, that is, the option is enabled by design.

The internal representations, organization, and procedures utilized by implementations when performing arithmetic operations (e.g., addition and subtraction) on numerical attributes (e.g., Integer data type) need only suffice to achieve the level of precision required to meet their designated performance levels. Designated performance levels can be imposed by a requirement within this standard (e.g., within procedures involved in determining the designated values of specified numerical attributes), and they may be imposed by a PTP Profile or the needs of the target application.

When implementing arithmetic operations, implementers must observe and correct for differences in data type, units, and in some cases scaling, for example, differences between data types of timestamps and correction fields.

Management configuration of numerical attributes that will be later used in arithmetic operations may be limited to a dynamic range smaller than the dynamic range enabled by the data type defined by the management interface of the attribute. The effective configurable dynamic range need only suffice to achieve the designated performance levels. Designated performance levels can be imposed by a requirement within this standard (e.g., within procedures involved in determining the designated values of specified numerical attributes), and they may be imposed by a PTP Profile or the needs of the target application.

6.1.2 Applicability and contents of the standard

The PTP standard specifies a clock synchronization protocol. This protocol is applicable to distributed systems consisting of two or more PTP Instances, communicating over a PTP Network. PTP Instances are modeled as containing a real-time clock that may be used by applications associated with the PTP Instance for various purposes, such as generating timestamps for data or ordering events managed by a PTP Instance. The protocol provides a mechanism for synchronizing the clocks of participating PTP Instances to a high degree of accuracy and precision. A PTP Node can contain multiple PTP Instances.

This standard specifies:

- a) The Precision Time Protocol
- b) The PTP Node, PTP Network, PTP Instance, and communication properties necessary to support PTP

6.1.3 Names for major entities defined by the standard

Numerous terms are defined in Clause 3 that refer to major entities specified by the standard. These are:

- a) PTP Instance
- b) PTP Node
- c) Boundary Clock
- d) Ordinary Clock
- e) Transparent Clock
- f) Local Clock, Local PTP Clock
- g) PTP Management Node

The relationships between these entities are illustrated in Figure 4.

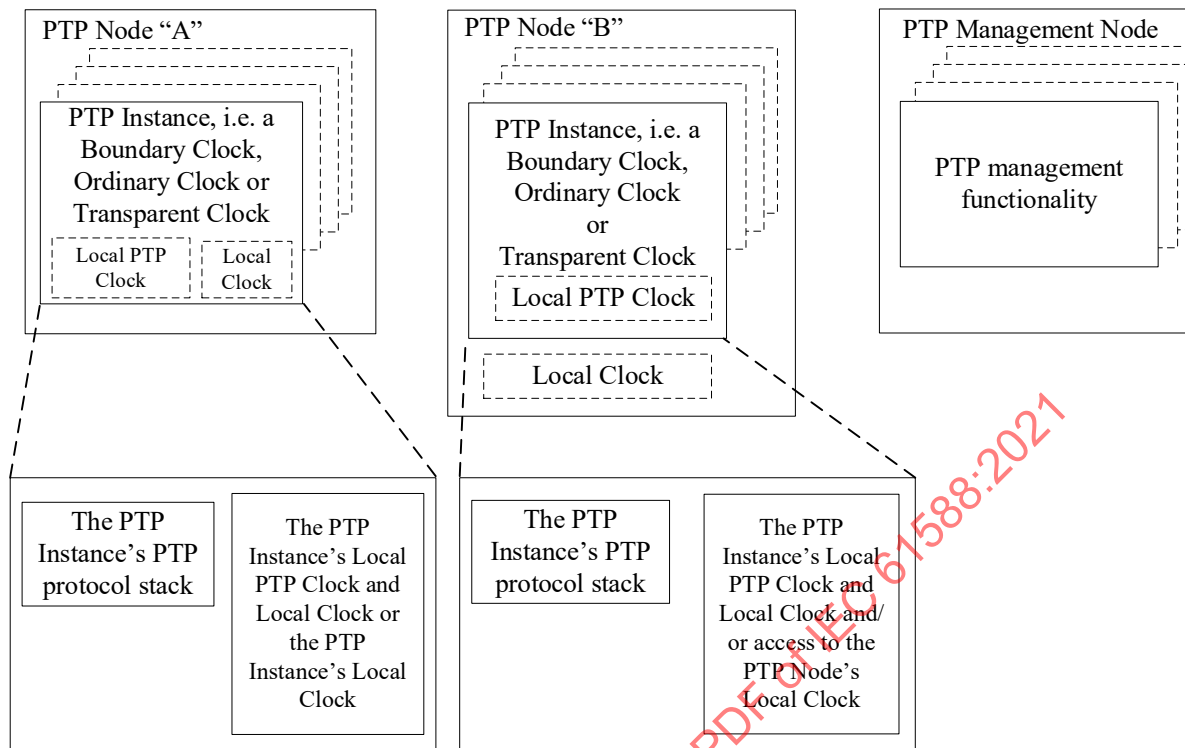


Figure 4 — Relationship of PTP entities

From Figure 4, each PTP Node can contain PTP Instances, that is, Boundary Clocks, Ordinary Clocks, or Transparent Clocks. For a single domain per 7.1, each such PTP Instance implements the functionality specified for the specific PTP Instance, that is, a Boundary Clock, Ordinary Clock, or Transparent Clock. For all PTP Instances, this functionality includes the PTP protocol stack specified for the type of the PTP Instance.

PTP Node "A" of Figure 4 illustrates PTP Instances, each of which contains either both a Local PTP Clock and a Local Clock or else only a Local Clock. The exact configuration depends on application requirements. For example, an Ordinary Clock will contain a Local PTP Clock and a Local Clock. By contrast, a Transparent Clock only needs a Local Clock although a Local PTP Clock can be present.

PTP Node "B" of Figure 4 illustrates PTP Instances, each of which contains a Local PTP Clock and/or access to a Local Clock common to all PTP Instances of the PTP Node. The exact configuration depends on application requirements. For example, an Ordinary Clock will contain a Local PTP Clock and have access to the Local Clock. By contrast, a Transparent Clock only needs the Local Clock although a Local PTP Clock can be present.

A PTP Management Node is a device in the PTP Network that uses PTP management messages to configure and/or monitor PTP Instances within PTP Nodes. A PTP Management Node contains the functionality specified in Clause 15. A PTP Management Node transmits PTP management messages with actions GET, SET, and/or COMMAND, and it receives PTP management messages with actions RESPONSE and/or ACKNOWLEDGE.

NOTE—Manufacturers of devices supporting the PTP protocol typically advertise these devices as "Boundary Clocks", "Ordinary Clocks", "Transparent Clocks", or "Grandmaster Clocks". These and other product designations are terms outside of this standard. Thus, for example, the term "Boundary Clock" in this standard has a specific meaning defined in the standard and does not necessarily describe a specific "Boundary Clock" product, and absent conformance certification, there is no assurance that a product named a "Boundary Clock" implements the specifications of this standard.

6.2 Principal assumptions about the network and implementation recommendations

The following principal assumptions and recommendations should be followed to ensure correct operation of the protocol. These are discussed in greater detail in later clauses:

- a) PTP assumes that the network eliminates cyclic forwarding of PTP messages within each communication path (e.g., by using a spanning tree protocol). However, the PTP protocol also eliminates cyclic forwarding of PTP messages.
- b) PTP is tolerant of an occasional missed PTP message, duplicated PTP message, or PTP message that arrived out of order. However, PTP assumes that such impairments are rare.
- c) PTP was designed assuming a multicast communication model. PTP also supports a unicast communication model, and a mixed multicast/unicast communication model, if the behavior of the protocol is preserved. PTP assumes that Announce messages are periodically sent by one PTP Port and delivered to all other PTP Ports within a PTP Communication Path. If the PTP Communication Path consists of more than two PTP Ports, the assumption is that Announce messages are either sent in multicast or the Announce information is replicated to all PTP Ports in the PTP Communication Path using unicast PTP messages. PTP Ports discover other ports within a PTP Communication Path through the receipt of multicast Announce messages. When multicast communication is not available, another form of discovery (e.g., by configuration) is required; see, for example, 17.4. PTP management messages sent to all PTP Ports also require either multicast messaging or replication of the PTP management message to all PTP Ports within the PTP Communication Path.
- d) If two-step PTP Ports are used, then the PTP Network must be designed such that the general message takes the same path (Sync and Follow_Up), or in some cases the reverse path (Delay_Req and Delay_Resp), as the event message through a Transparent Clock. Failure to do this will result in a condition where PTP does not calculate path delay properly. This condition is undetectable and may introduce additional jitter and wander, but it will not cause the operation of the PTP protocol to cease.
- e) PTP assumes that the number of Boundary Clocks forming the master–slave synchronization hierarchy from the Grandmaster PTP Instance to any Slave PTP Instance is less than 255 (see 9.3.2.5).
- f) Network components that are not PTP-aware, for example, bridges, introduce timing jitter and wander that, if uncorrected, can degrade time transfer accuracy. Since the jitter and wander are often traffic dependent, the network traffic patterns should be designed to minimize the traffic and to minimize the variation in the traffic load. It is also recommended that PTP event messages be sent in high priority compared with other data (see A.5.3.3). Whenever possible such network components should be replaced by PTP Boundary Clocks or Transparent Clocks.
- g) The underlying transport protocol is structured such that a message timestamp point can be defined.

6.3 PTP Networks

The PTP Instances in a PTP Network execute the PTP protocol to synchronize the clocks in the PTP Instances.

The protocol executes within a logical scope called a “domain” as specified in 7.1. Unless otherwise specified, all PTP messages, data sets, state machines, and all other PTP entities are always associated with a particular domain. A PTP Network can support multiple domains. Within this standard, the time established within one domain by the protocol is independent of the time in other domains.

The PTP protocol is a distributed protocol that specifies how the real-time clocks in the PTP Instances of each domain of the PTP Network synchronize with each other. The PTP Instances containing these clocks

are organized into a master–slave synchronization hierarchy whereby the PTP Instance at the top of the hierarchy—the Grandmaster PTP Instance—determines the reference time for the entire domain. The synchronization is achieved by exchanging PTP timing messages with the Slave PTP Instances using the timing information to adjust their clocks to the time of their respective Parent PTP Instances in the hierarchy.

PTP Instances in a PTP Network communicate with each other via a communication network. The PTP Network can include translation devices between segments implementing different network communication protocols.

6.4 PTP message classes

The protocol defines event and general PTP messages. PTP event messages are timed messages in that an accurate timestamp is generated at both transmission and receipt as described in 6.6.5. PTP general messages do not require accurate timestamps.

The set of PTP event messages consists of:

- a) Sync (see 13.6)
- b) Delay_Req (see 13.6)
- c) Pdelay_Req (see 13.9)
- d) Pdelay_Resp (see 13.10)

The set of PTP general messages consists of:

- Announce (see 13.5)
- Follow_Up (see 13.7)
- Delay_Resp (see 13.8)
- Pdelay_Resp_Follow_Up (see 13.11)
- Management (see Clause 15)
- Signaling (see 13.12)

PTP timing messages are the set of all PTP event messages plus the following PTP general messages: Follow_Up, Delay_Resp, and Pdelay_Resp_Follow_Up.

The Sync, Delay_Req, Follow_Up, and Delay_Resp messages are used to generate and communicate the timing information needed to synchronize Ordinary Clocks and Boundary Clocks using the delay request–response mechanism.

The Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages are used to measure the PTP Link delay between two PTP Ports implementing the peer-to-peer delay mechanism. The PTP Link delay is used to correct timing information in Sync and Follow_Up messages transmitted over PTP Links. Ordinary Clocks and Boundary Clocks that implement the peer-to-peer delay mechanism can synchronize using the measured PTP Link delays and the information in the Sync and Follow_Up messages.

The Announce message is used to establish the synchronization hierarchy.

The PTP management messages are used to query and update the PTP data sets maintained by PTP Instances. These messages are also used to configure a PTP Network and for initialization and fault management. PTP management messages are used between PTP Management Nodes and PTP Instances.

NOTE—PTP Nodes can support other management mechanisms as an alternative to PTP management messages (see 8.1.4.1).

The Signaling messages are used for communication between PTP Instances for all other purposes. For example, Signaling messages can be used for negotiation of the rate of unicast PTP messages between a Master PTP Instance and its Slave PTP Instances.

All PTP Messages can be extended by means of a standard type, length, value (TLV) extension mechanism. For example, the PATH_TRACE message extensions can be used to detect rogue Announce messages. See 16.2.1 for more detail on rogue Announce messages.

6.5 PTP device types

6.5.1 General

There are five basic types of PTP devices, as follows:

- a) Ordinary Clock
- b) Boundary Clock
- c) End-to-end Transparent Clock
- d) Peer-to-peer Transparent Clock
- e) PTP Management Node (6.1.3)

All five types implement one or more aspects of the protocol.

Two mechanisms are used in PTP to measure the propagation delay between PTP Ports. The first one, the delay request-response mechanism, uses the messages Sync, Delay_Req, Delay_Resp, and if required, Follow_Up (see 11.3). The second one, the peer-to-peer delay mechanism, uses the messages Pdelay_Req, Pdelay_Resp, and if required, Pdelay_Resp_Follow_Up (see 11.4). PTP Ports on Ordinary Clocks and Boundary Clocks can be implemented using either mechanism. PTP Ports on end-to-end Transparent Clocks participate in the delay request-response mechanism. PTP Ports on peer-to-peer Transparent Clocks use the peer-to-peer delay mechanism. These two mechanisms do not interwork on the same PTP Communication Path. In addition, the peer-to-peer delay mechanism is restricted to topologies where each peer-to-peer PTP Port communicates PTP messages with at most one other peer-to-peer PTP Port (see 11.4.3).

The use of the various PTP Instance types is limited as follows:

- Ordinary Clocks and Boundary Clocks with PTP Ports implementing only the peer-to-peer delay mechanism, and peer-to-peer Transparent Clocks, can only be connected in topologies where each such PTP Port implementing the peer-to-peer delay mechanism communicates PTP messages to and from a single PTP Port also implementing the peer-to-peer delay mechanism (see 11.4.3). Except in carefully designed networks, this will preclude the use of end-to-end Transparent Clocks and bridges that do not support PTP, for example, a conventional bridge, when using the peer-to-peer delay mechanism.
- Ordinary Clocks and Boundary Clocks with PTP Ports implementing only the delay request-response mechanism, and end-to-end Transparent Clocks, can be connected in any topology that excludes other PTP Ports using the peer-to-peer delay mechanism. This precludes the use of peer-to-peer Transparent Clocks in such a PTP Network.
- A Boundary Clock with PTP Ports supporting each of the two mechanisms may be used to bridge between regions supporting the different mechanisms.

6.5.2 Ordinary Clocks and Boundary Clocks

6.5.2.1 Layered model of Ordinary Clocks and Boundary Clocks

6.5.2.1.1 General

Figure 5 illustrates the layered model of an Ordinary Clock, whereas Figure 6 illustrates the layered model of a Boundary Clock.

The purpose of these models is to clarify the relationships and functions of the clauses of this standard. These models are not to be construed as implementation specifications.

NOTE—The models of Figure 5 and Figure 6 are restatements of Figures 2 and 3 of IEEE Std 1588-2008 except for the model of timestamp generation, which has been moved to 6.5.2.1.4. With the exception of the Special Port for media not using PTP timing messages, that is, the rightmost PTP Port in Figure 6, these models are compatible with the models used in IEEE Std 1588-2008.

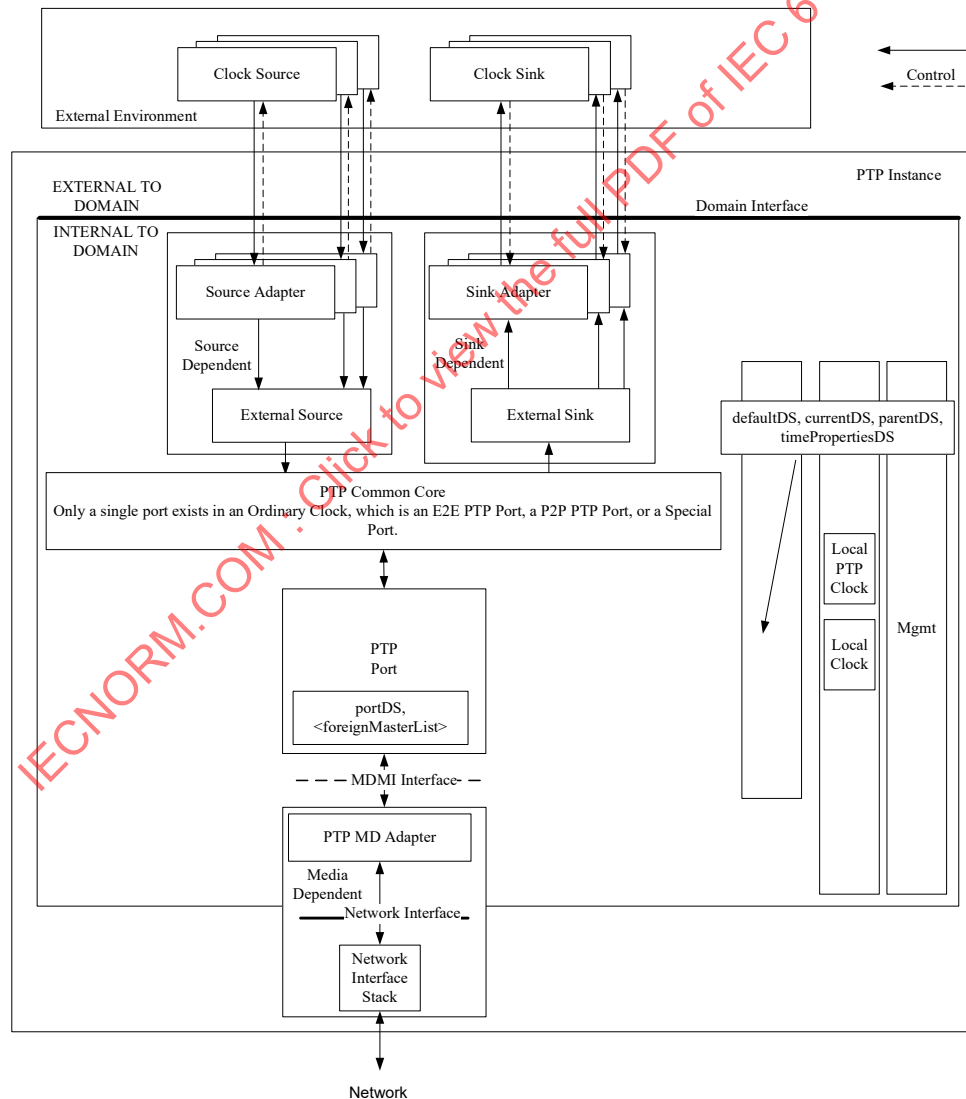


Figure 5—Layered model of an Ordinary Clock

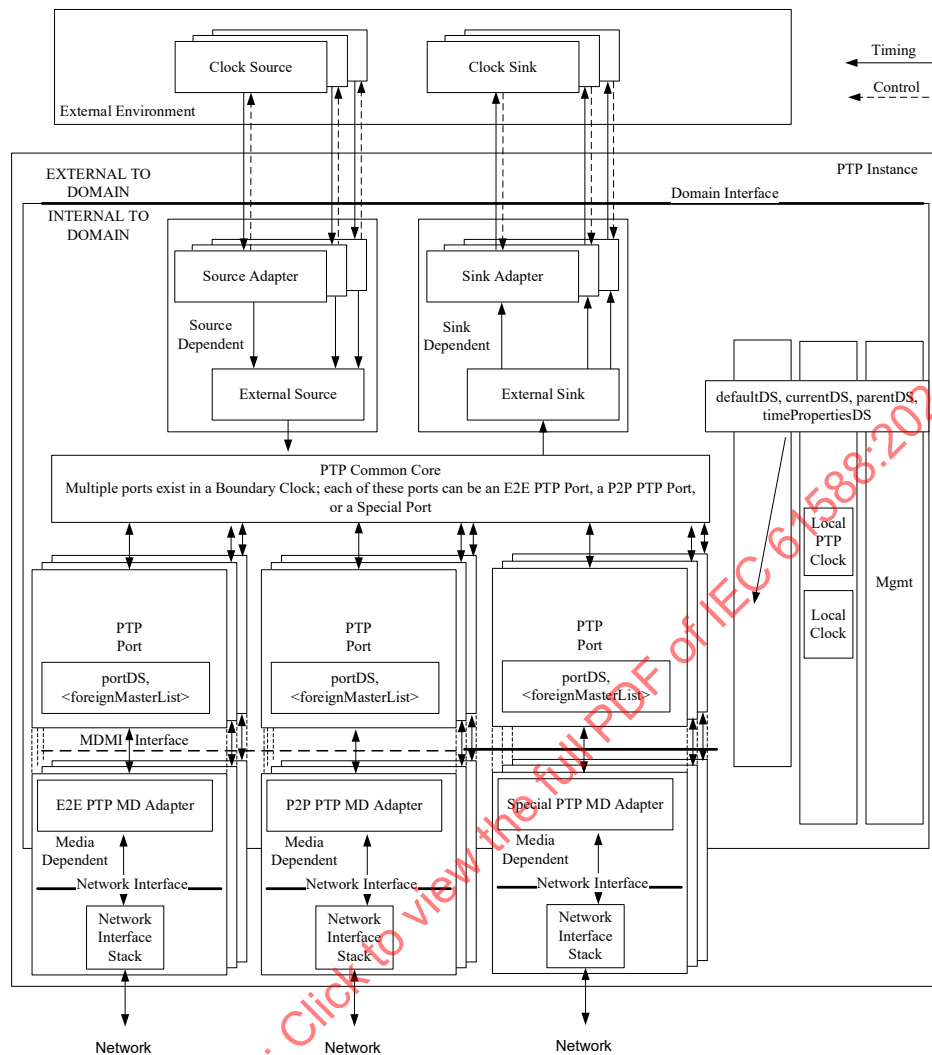


Figure 6—Layered model of a Boundary Clock

Differences between Ordinary Clocks and Boundary Clocks are elaborated in 6.5.2.2 and 6.5.2.3, respectively. The principal features of the models illustrated in Figure 5 and Figure 6 are as follows.

Each figure is divided into three regions:

- a) The external environment shown at the top of the figures. The main goal of PTP is to transfer time from an application-specific source of time to application devices. As noted in 6.3, the PTP protocol executes within a logical scope called a domain. The external environment is not within the domain as indicated by the solid line annotated as external and internal to the domain. Contained in the external environment are
 - 1) Clock Sources: External sources of the time to be distributed by PTP, for example, GNSS receivers.
 - 2) Clock Sinks: Users of the PTP-distributed time. Typical users are sensors, actuators, and computation elements.
- b) PTP Instance.
- c) Communication network over which a PTP Instance communicates with its peers in the domain.

6.5.2.1.2 External environment and the Source Dependent and Sink Dependent blocks

6.5.2.1.2.1 Introducing time into a PTP Instance

Time from one or more external Clock Sources, that is, Clock Source blocks, can be introduced into a PTP Instance (and the domain) only via Source Dependent blocks. This is a one-way transfer of time from the external environment into the PTP Instance, that is, the operation of PTP does not change the time maintained by the external Clock Source. This transfer can require timing control signals as illustrated.

Typical external Clock Sources are GNSS receivers or terrestrial radio links. See Table 6 for an enumeration used to characterize such sources. Source Adapter blocks manage the differences between an interface unique to the external source and a common implementation-specific interface to the External Source block. The External Source block is responsible for selecting which of the external Clock Sources, or if none are available an internal source, is to be used as the basis for assigning values to the members of the default data set. In the event that the PTP Instance is selected as the Grandmaster PTP Instance, the External Source block also provides data to populate the time properties data set. See 7.6.6 for specifications on updating these data sets.

6.5.2.1.2.2 Providing time to an external application

Time from a PTP Instance can be provided to an external application, that is, Clock Sink block, only via the Sink Dependent block (see 7.6.7). This is a one-way transfer of time from the PTP Instance to external Clock Sinks; that is, the operation of PTP is not changed as a result of this time transfer. This transfer can require timing control signals as illustrated. There can be multiple sinks served by a PTP Instance.

Typical external applications, that is, Clock Sinks, are sensors, actuators, and waveform generators. Sink Adapter blocks manage the differences between an interface unique to the external Clock Sink and a common implementation-specific interface to the External Sink block. The External Sink block is responsible for delivery of either PTP- or ARB-based time (see 7.2.1), as well as for accompanying meta-data to this interface.

6.5.2.1.3 Common core, data sets, internal clocks, and management

6.5.2.1.3.1 PTP Common Core block

The PTP Common Core block is responsible for all operations of the PTP protocol that are common to the other blocks to the PTP Instance as a whole, as well as for coordinating the activities of the other blocks in the PTP Instance. These responsibilities include the following:

- a) Operations, except for management (see Clause 15), of the default, parent, current, time properties, and optional data sets that are not per PTP Port (see Clause 8)
- b) Aspects of state configuration, including the BMCA, that require considerations of information from all PTP Ports (see Clause 9)
- c) Aspects of PTP message handling that require considerations of information from all PTP Ports, and inter-port PTP message and information handling for retransmission (see Clause 10 and Clause 11)
- d) Aspects of optional PTP features that require consideration of information from all PTP Ports (see Clause 16 and Clause 17, Annex J, and Annex P)

6.5.2.1.3.2 Clocks associated with a PTP Instance

The PTP protocol maintains time within each PTP Instance and within the domain, based on either the timescale PTP or the timescale ARB (see 7.2.1). The protocol can distribute information enabling an external Clock Sink to convert from one of these timescales to an application-specific timescale. This conversion can be done in the Sink Adapter block or by the external Clock Sink itself.

The timescale PTP or the timescale ARB is maintained in the Local PTP Clock block. The time of Local PTP Clock represents the local estimate of the time of the Grandmaster Clock, that is, the time of the Local PTP Clock of the Grandmaster PTP Instance of the domain. There are two methods of maintaining the Local PTP Clock:

- 1) Maintain an actual physical clock, that is, a local oscillator and a time counter, synchronized to the time of the Grandmaster Clock, or in the case of the Grandmaster Clock to the selected external time source represented in Figure 5 and Figure 6 by the Clock Source block. In this method, the Local PTP Clock is identical to the Local Clock.
- 2) Maintain a model that computes the timescale PTP or the timescale ARB based on the Local Clock. In this method, the model incorporates the results of the synchronization procedures.

The synchronization procedures are discussed in Clause 11 and Clause 12.

All blocks in the PTP Instance can obtain time from these clocks.

6.5.2.1.3.3 Management

The Management block (labeled “Mgmt” in Figure 5 and Figure 6) can communicate with all blocks in the PTP Instance. The Management block allows observing and, in some cases, configuring aspects of the PTP data sets and the operation of the protocol. Various management mechanisms can be applied to the Management block (see 8.1.4.1).

PTP management messages (see Clause 15) use the targetPortIdentity field of each message to identify the specific PTP Instance (and PTP Port) that is managed by the message. Therefore, for PTP management messages, the Management block operates within each PTP Instance.

Management mechanisms (e.g., SNMP) other than PTP management messages are typically implemented in the PTP Node. Following the organization specified in 8.1.4.2, these management mechanisms provide a data model for access to each PTP Instance (and PTP Port). Therefore, for these management mechanisms, the Management block operates within the PTP Node, but it provides access to each PTP Instance through its data model.

6.5.2.1.4 PTP Ports and the PTP Network

The models of this subclause apply to Boundary Clocks, Ordinary Clocks, and Transparent Clocks, except that Transparent Clocks do not have Special Ports.

Each PTP Port consists of a PTP Port block and a Media Dependent block.

The PTP Port block is separated from the Media Dependent block by the MDMI interface. The MDMI interface must be supported for Special Ports and can be supported for E2E and P2P PTP Ports. The MDMI interface for Special Ports is specified in 11.5.

NOTE 1—The MDMI interface is new to this edition and is introduced to facilitate the use within the PTP protocol of media that implement their own methodology for measuring path delay and transferring time, for example, IEEE Std 802.11.

The PTP Port block is the media-independent portion of the PTP Port that is responsible for all per-port functionality that does not depend on the communication media or the time transfer method, for example, E2E, P2P or Special, being used. These responsibilities include:

- Operations and maintenance of the PTP Port data sets and the <foreignMasterList> (see 8.2.15 and 9.3.2.4, respectively), and any optional data sets, if present, that are per PTP Port
- Operation of the per-port portion of the BMCA (see 9.3)

The Media Dependent block is the media-dependent portion of the PTP Port that consists of two sub-blocks: a PTP MD Adapter and a Network Interface Stack. The PTP MD Adapter block will differ depending on the time transfer methodology used, that is, E2E, P2P, or Special.

The MD Adapters and Network Interface Stack blocks are separated by the Network Interface. The PTP MD Adapter is responsible for media-dependent functions of the PTP protocol, including:

- Handling the transmission and reception of all PTP messages
- Handling the transfer of time between the Network Interface Stack and the PTP Port block

The PTP MD Adapter block communicates with the Network Interface Stack via three interfaces: General Message Functions, Event Message Functions, and Other Adapter Functions, as illustrated in Figure 7, all on a single physical port.

The Event Message Function block is used to send and receive PTP event messages, which are timestamped by the Timestamp Generation block based on the value of the Timestamping Clock. To achieve the best accuracy and least jitter, the detection of PTP event messages and subsequent capture of the time of the Timestamping Clock is preferably implemented as close to the network as possible. Many of the modern PHY chips therefore generate a signal upon detecting a PTP event messages. This signal is then used to capture the timestamp. Therefore, in Figure 7, the detection of PTP event messages and the generation of the timestamp can occur either in the MD Adapter block or, preferably, in the Network Interface Stack as illustrated. See 6.6.5 for a more complete discussion of timestamp generation. In either case, the MD Adapter block is responsible for transferring both the information contained in the PTP event message as well as the timestamp to the PTP Port block.

The General Message Function block is used to send and receive PTP general messages. The Other Adapter Functions block handles all other functions and any coordination of the other blocks of the MD Adapter block.

There are three forms of the PTP Port:

- a) The end-to-end (E2E) form consists of a PTP Port block and a corresponding media-dependent E2E PTP MD Adapter block. These blocks implement the end-to-end methodology of time transfer and determining path delay (see 11.3). These blocks make use of PTP messages communicated over the PTP Network to other PTP Instances in the domain to implement two-way time transfer between PTP Instances.
- b) The peer-to-peer (P2P) form consists of a PTP Port block and a corresponding media-dependent P2P PTP MD Adapter block. These blocks implement the peer-to-peer methodology of time transfer and determining path delay (see 11.4). These blocks make use of PTP messages communicated over the PTP Network to other PTP Instances in the domain to implement two-way time transfer between PTP Instances.
- c) The Special form consists of a PTP Port block and a corresponding media-dependent Special PTP MD Adapter block. These blocks implement time transfer over media using media-specific time transfer methods (not PTP timing messages) to transfer time between PTP Instances in a domain (see 11.5, which specifies the details of the Special Ports including the MDMI interface).

NOTE 2—The Special Port allows the use of media-specific time transfer methods specified by media standards organizations, for example, IEEE Std 802.11. See 11.5.

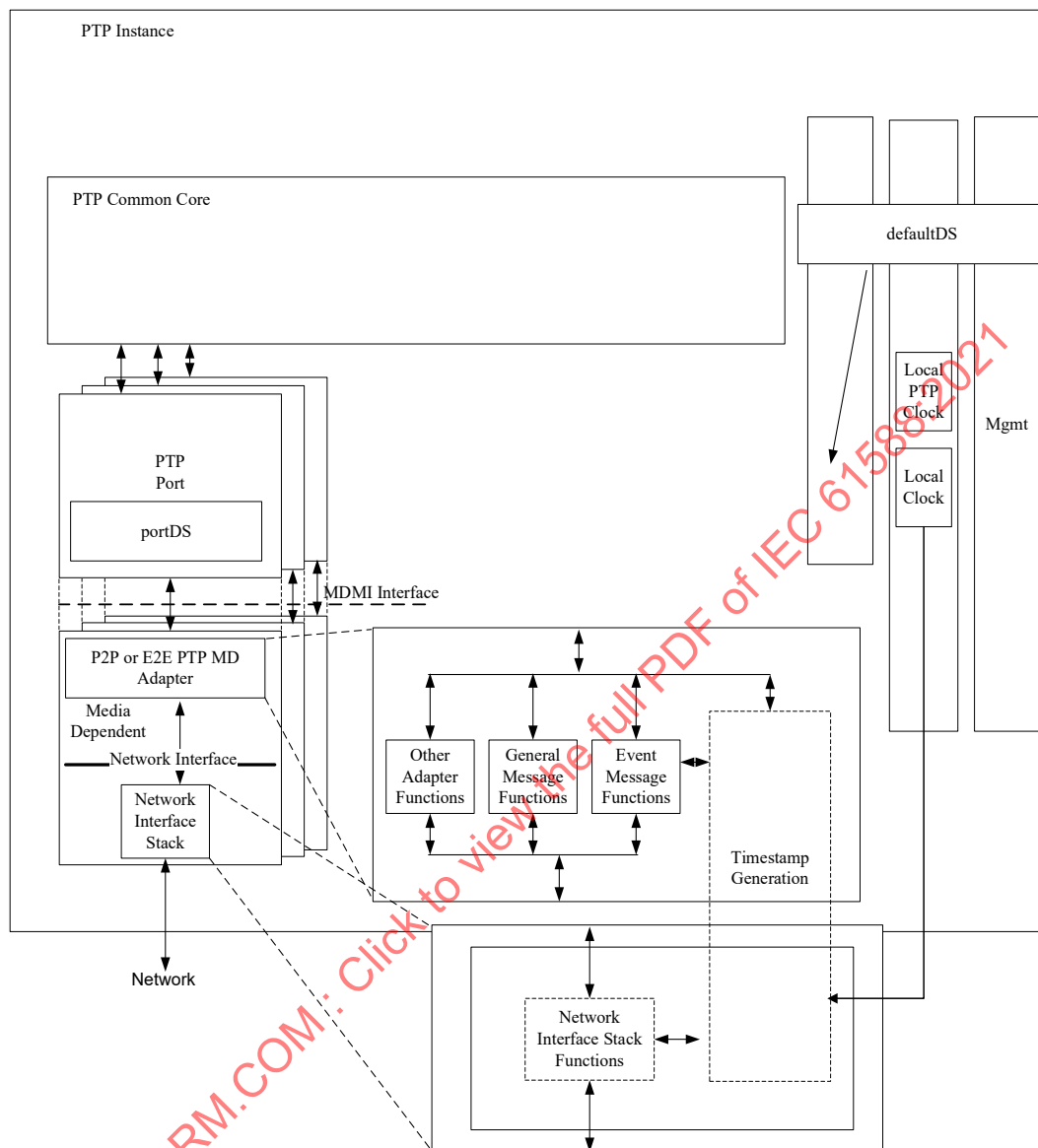


Figure 7—Model of the message interfaces of the MD Adapter block

6.5.2.2 Properties of an Ordinary Clock

An Ordinary Clock in a domain supports a single copy of the protocol and has a single PTP Port (see Figure 5). This single PTP Port can be an E2E, a P2P, or a Special Port see (6.5.2.1.4). The Ordinary Clock can be the Grandmaster PTP Instance, or it can be a Slave PTP Instance in the master–slave hierarchy.

In some applications, such as industrial automation, an Ordinary Clock can also be associated with an application device such as a sensor or an actuator. In a telecommunications application, an Ordinary Clock can be associated with a timing demarcation device. Using the Sink Dependent blocks, the Local PTP Clock provides time support to one or more such a device as illustrated in Figure 5.

6.5.2.3 Properties of a Boundary Clock

The Boundary Clock has several PTP Ports (see Figure 6). Each PTP Port of a Boundary Clock is like the PTP Port of an Ordinary Clock.

All PTP messages except PTP management messages are terminated at a Boundary Clock and are not retransmitted. PTP management messages are retransmitted by other PTP Ports on the Boundary Clock subject to restrictions to limit the propagation of these PTP messages within the PTP Network.

The Boundary Clock model of Figure 6 is applicable only to PTP messages and non-PTP messages used by the Special Port.

A Boundary Clock can become the Grandmaster PTP Instance of the domain.

6.5.3 End-to-end Transparent Clocks

The model of an end-to-end Transparent Clock is illustrated in Figure 8.

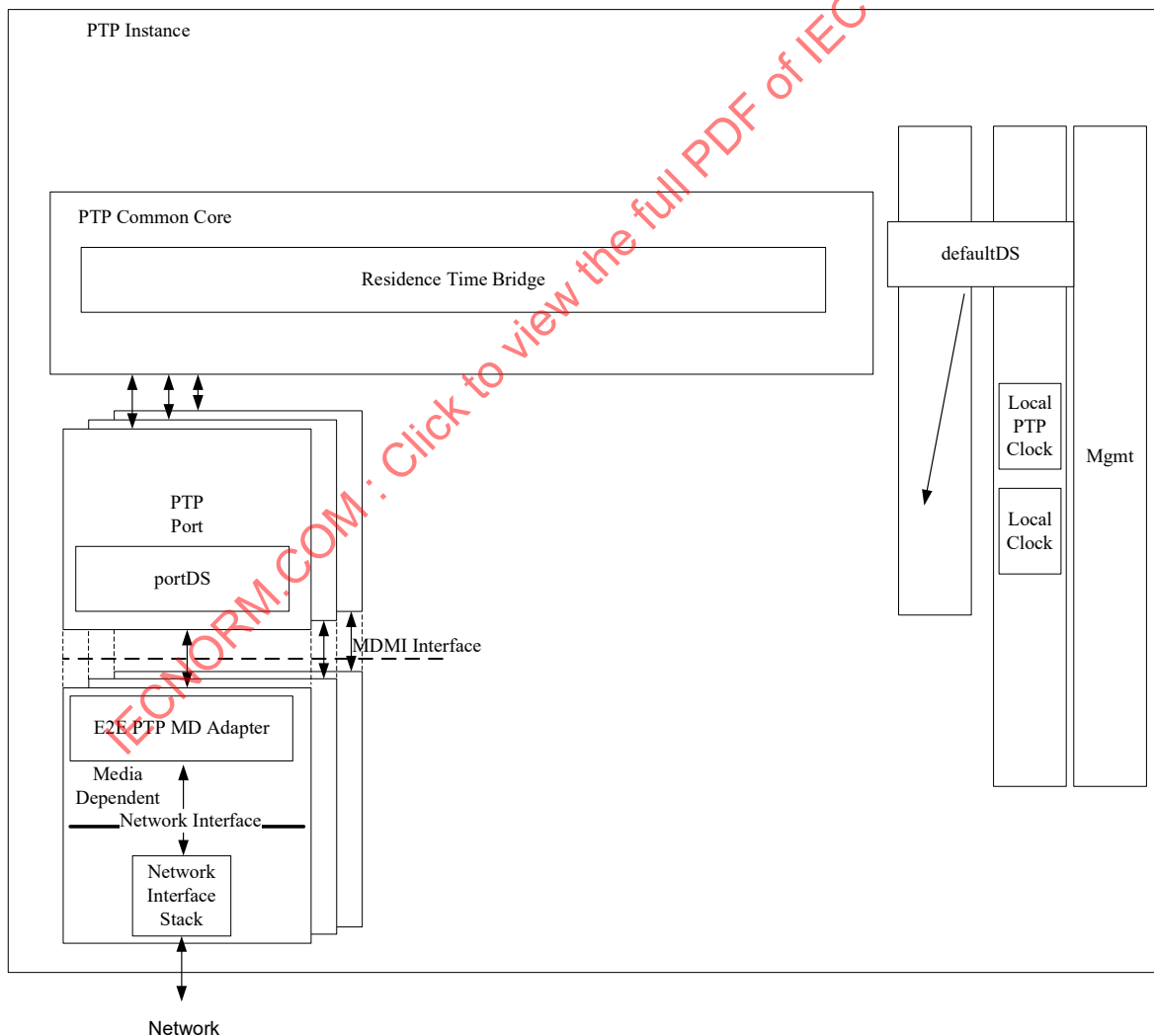


Figure 8—Layered model of an End-to-end Transparent Clock

The end-to-end Transparent Clock handles all messages as specified in 7.3.1.

Unlike Boundary Clocks or Ordinary Clocks, the PTP Ports on a Transparent Clock do not maintain PTP Port state, for example, MASTER or SLAVE. Instead ingress PTP messages received on a Transparent Clock's PTP Port are retransmitted on all other PTP Ports of the Transparent Clock subject to the rules of the underlying transport protocol.

The Residence Time Bridge, shown in Figure 8, measures the residence time of PTP event messages (the time the message takes to traverse the Transparent Clock). These residence times are accumulated in a special field, the correctionField, of the PTP event message or the associated follow-up message (Follow_Up or Pdelay_Resp_Follow_Up). This correction is based on the difference in the timestamps generated when the PTP event message enters and leaves the Transparent Clock. Any updates to checksums and other protocol fields required by the network protocol are made.

Note that while at the PTP layers (i.e., above the Network Interface; see Figure 8) the retransmitted PTP message is a copy or modification of the received PTP message, the transmitted PDU of the Network Interface Stack (i.e., below the Network Interface) is a new PDU, different from the PDU that contains the received PTP message. For example, in the case of transport over full-duplex Ethernet, the transmitted Ethernet frame is a different Ethernet frame from the received Ethernet frame, even though only the correction field of the PTP message contained in the transmitted frame has been modified.

Note that the value of the correction update is specific to each egress PTP Port and PTP message since the residence times are not necessarily the same for all paths through the Transparent Clock or for successive PTP messages on the same path. The correction process is illustrated for an arbitrary pair of PTP Ports in Figure 9. This accumulated measure of residence times allows downstream Ordinary Clocks or Boundary Clocks to correct for the residence time portion of the PTP Communication Path delay.

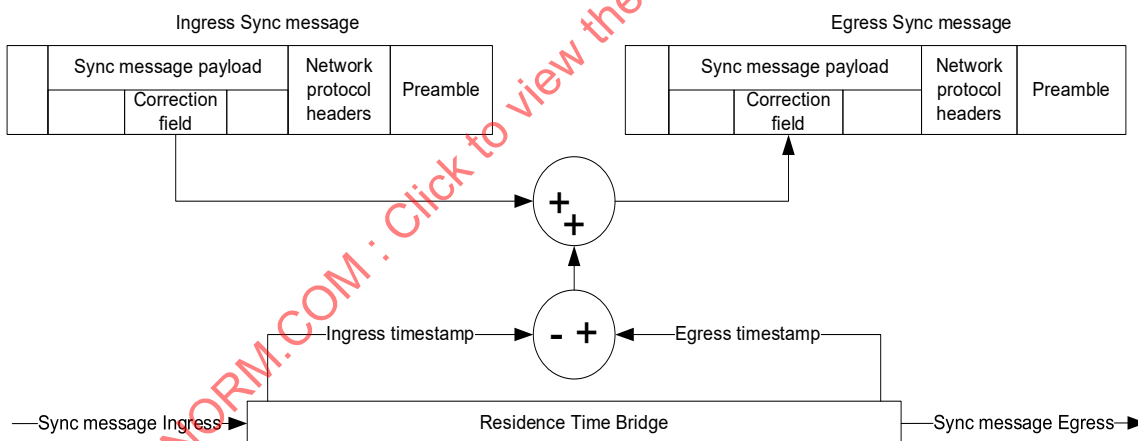


Figure 9—One-step end-to-end residence time correction for Sync message processing

Since only residence times are measured by a Transparent Clock, only a Local Clock, and not a Local PTP Clock need be maintained.

The timestamps used in computing the residence time are based on timestamps generated from the Timestamping Clock. Since these accumulated residence times are used by a Slave PTP Instance to adjust the time provided by a Master PTP Instance, it is important that any errors resulting from differences in the rates of the Timestamping Clock of the Master PTP Instance and the Timestamping Clock of the Transparent Clocks be negligible for the accuracy required by the application. For example, when using a 100 ppm clock, it is possible that the rates of the two clocks (essentially the definition of the second on the two clocks) can differ by 0.02%, and the error introduced can be 0.02% of the measured residence time.

Thus, for a residence time of 1 ms, the maximum error is 200 ns. This error might be unacceptably large, but it can be reduced by correcting for the differences between the rate of the Timestamping Clock and the rate of the Grandmaster Clock by syntonizing the Timestamping Clock to the Grandmaster Clock, or by using a Local PTP Clock.

NOTE—Implementers of PTP Nodes need to consider the resources required to support multiple domains. The inability to process the protocol in a timely fashion due to resource limitations can lead to deterioration in the synchronization performance, thrashing, or failure of the protocol. Users need to be aware of this limitation when selecting PTP Nodes and designing their systems.

6.5.4 Peer-to-peer Transparent Clocks

The model of a peer-to-peer Transparent Clock is illustrated in Figure 10.

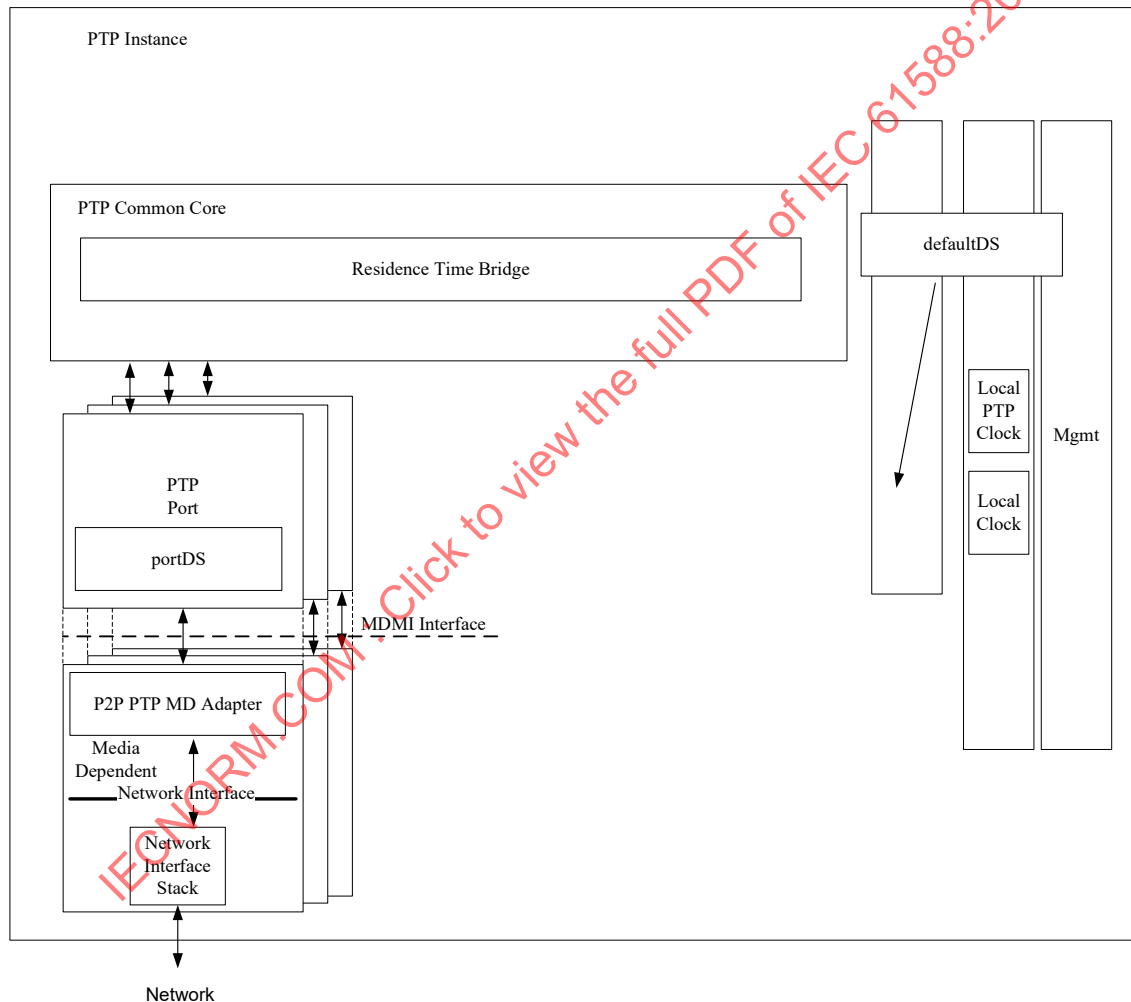


Figure 10—Layered model of a peer-to-peer Transparent Clock

The peer-to-peer Transparent Clock differs from the end-to-end Transparent Clock in the way it corrects and handles PTP timing messages. In all other respects, it is identical to the end-to-end Transparent Clock.

The PTP Ports of a peer-to-peer Transparent Clock compute the PTP Link delay between each PTP Port and a similarly equipped PTP Port on another PTP Instance sharing the PTP Link, that is, the PTP Link

peer. The PTP Link peer exists in another PTP Instance supporting the peer-to-peer delay mechanism since non-peer-to-peer PTP Instances are not expected on a PTP Link connected to a peer-to-peer Transparent Clock. The computation of PTP Link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and possibly Pdelay_Resp_Follow_Up messages with the PTP Link peer. As a result of these exchanges, the PTP Link delay is known for each PTP Port of the peer-to-peer Transparent Clock. Unlike the end-to-end Transparent Clock, which updates and retransmits all PTP timing messages, the peer-to-peer Transparent Clock only updates and retransmits Sync and Follow_Up messages. These transmissions are as specified in 7.3.1. The correctionField in the Sync or Follow Up message is updated for both the residence time of the Sync message within the peer-to-peer Transparent Clock and the PTP Link delay measured on the Transparent Clock's PTP Port receiving the Sync message.

This correction process is illustrated in Figure 11.

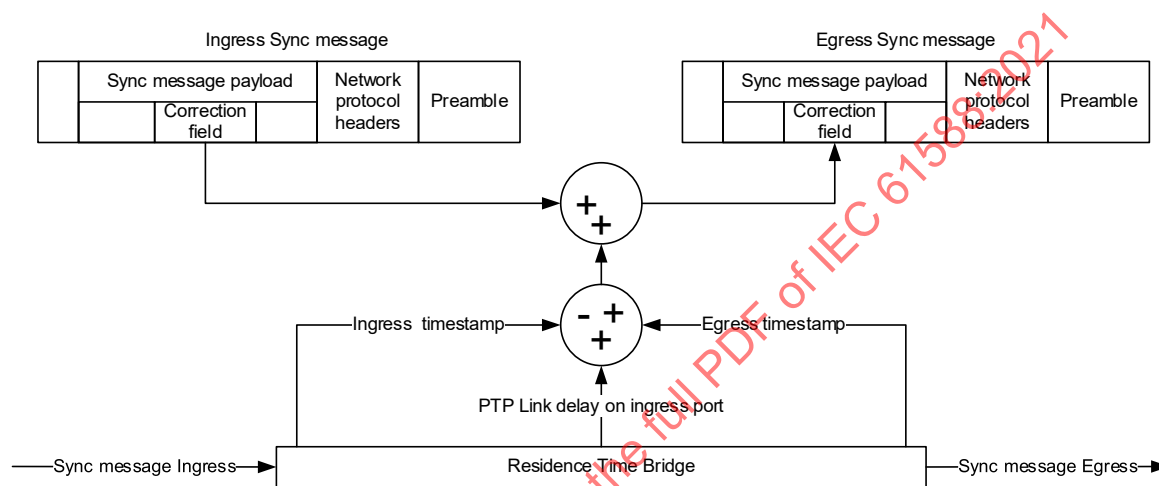


Figure 11—One-step peer-to-peer residence time and PTP Link delay corrections for Sync message processing

The egress PTP timing message contains the residence time and PTP Link delay corrections for the actual ingress PTP Link traversed by a Sync message. A change in Sync message routing through a PTP Network of peer-to-peer PTP Instances might result in the Sync message entering the peer-to-peer Transparent Clock from a different PTP Link. Since the peer-to-peer Transparent Clock corrects for the residence and PTP Link delays based on the actual PTP Link and internal path through the PTP Instance taken by each Sync message, the egress timing information is always correct to within the accuracy of the measurements. Therefore, the timing information provided to the Slave PTP Instance always reflects the actual PTP Communication Path delay through the PTP Network of peer-to-peer Transparent Clocks. By contrast, in the case of the end-to-end corrections, the Slave PTP Instance waits for a new delay value based on the exchange of timing messages in the delay request-response mechanism, which takes more time, since the PTP messages need to traverse the entire new PTP Communication Path before correct values are present at the Slave PTP Instance.

The measurement of delay using the peer-to-peer delay mechanism does not interwork with delay measurements based on the delay request-response mechanism. As a result, a peer-to-peer Transparent Clock can only work with PTP Ports supporting the peer-to-peer delay mechanism. Likewise, end-to-end Transparent Clocks support only the delay request-response mechanism and not the peer-to-peer delay mechanism. If a network contains peer-to-peer Transparent Clocks in one region and end-to-end Transparent Clocks in another region, the regions can be connected only through a Boundary Clock.

The timestamps used in computing the residence and PTP Link delays are based on timestamps generated using the Timestamping Clocks. Since these accumulated residences and PTP Link delays are used by a Slave PTP Instance to adjust the time provided by a Master PTP Instance, it is important that any errors

resulting from differences between the rates of the Master Clock and the Timestamping Clock of each of the Transparent Clocks be negligible for the accuracy required by the application. For example, when using a 100 ppm clock, it is possible that the rates of the two clocks (essentially the definition of the second on the two clocks) can differ by 0.02%, and the error introduced can be 0.02% of the measured residence time. Thus, for a residence time of 1 ms, the maximum error is 200 ns. This error might be unacceptably large, but it can be reduced by correcting for the differences between the rate of the Timestamping Clock in this Transparent Clock and the rates of the clocks in the other PTP Instances in the PTP Network.

6.5.5 Combined Transparent Clocks and Ordinary Clocks

A Transparent Clock can be used as a network element (e.g., switch or router), or it can be associated with application devices such as sensors or actuators. In the latter case, an Ordinary Clock is combined with the Transparent Clock to provide real-time support for the application device. A model of this combination is illustrated in Figure 12.

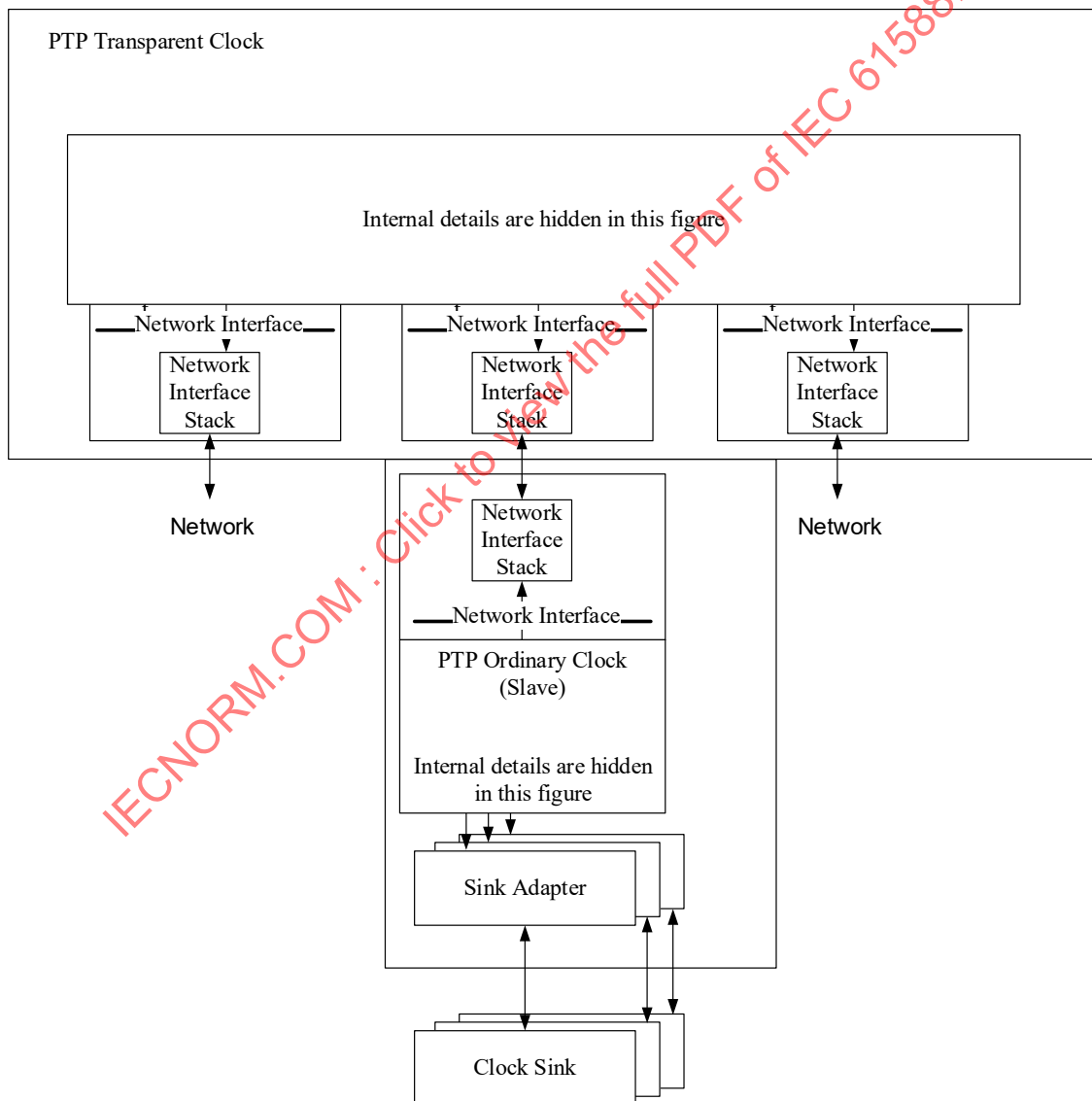


Figure 12—Layered model of a device that combines Transparent Clock and Ordinary Clock

The model of Figure 12 illustrates a three PTP Port Transparent Clock and an Ordinary Clock all in the same PTP Node. Two PTP Ports are normal Transparent Clock PTP Ports and are connected to the domain network. The third PTP Port of the Transparent Clock directly connects to the “internal” Ordinary Clock. In this case, the actual Network Interface Stacks for this connection would implement internal time transfer, for example, over board traces. Such a PTP Node is useful in long linear chains of combined PTP Instances with the two normal network PTP Ports connected in the chain and with a sensor or actuator associated with each Ordinary Clock.

6.5.6 PTP Management Nodes

A PTP Management Node is a device that:

- a) Has one or more physical connections to the PTP Network
- b) Serves as a human or programmatic interface to PTP management messages
- c) May be combined with any of the PTP Instance types.

NOTE 1—As described in 6.1.3, a PTP Management Node uses PTP management messages to configure and/or monitor the PTP Instances within PTP Nodes.

NOTE 2—An application that manages PTP Nodes can support other management mechanisms as an alternative to PTP management messages (see 8.1.4.1).

6.6 Synchronization overview

6.6.1 General

The following two phases occur in the normal execution of the protocol:

- Establishing the master–slave hierarchy
- Synchronizing the Local PTP Clocks

6.6.2 Establishing the master–slave hierarchy

6.6.2.1 General

Within a domain, the PTP Ports of all Ordinary Clocks and Boundary Clocks each execute an independent copy of the protocol state machine. For “state decision events,” each PTP Port examines the contents of all Announce messages received on the PTP Port. Using the best master clock algorithm, the Announce message contents and the contents of the data sets associated with the Ordinary Clock or Boundary Clock are analyzed to determine the state of each PTP Port of the PTP Instance.

6.6.2.2 PTP state machine

Each PTP Port of an Ordinary Clock or Boundary Clock maintains a separate copy of the PTP state machine. This state machine defines the allowed states of the PTP Port and the transition rules between states. The principal “state decision events” determining the master–slave hierarchy are the receipt of an Announce message and the end of an announceInterval (the interval between Announce messages).

The PTP Port states determining the master–slave hierarchy are as follows:

- a) **MASTER:** The PTP Port is the source of time on the PTP Communication Path served by the PTP Port.
- b) **SLAVE:** The PTP Port synchronizes to the PTP Port on the PTP Communication Path that is in the MASTER state.
- c) **PASSIVE:** The PTP Port is not the source of time on the PTP Communication Path nor does it synchronize to a Master Clock.

6.6.2.3 Best master clock algorithm

The best master clock algorithm compares data describing two PTP Instances to determine which data describe the better PTP Instance. This algorithm is used to determine which of the PTP Instances described in several Announce messages received by the PTP Ports of a PTP Instance is the best PTP Instance. It is also used to determine whether a newly discovered potential PTP Grandmaster Instance is better than the PTP Instance itself. The data describing a potential PTP Grandmaster Instance are contained in the grandmaster fields of an Announce message. The data describing the PTP Instance receiving the Announce message are contained in the defaultDS data set of the PTP Instance.

The best master clock algorithm is composed of two separate algorithms:

- Data set comparison algorithm
- State decision algorithm

The data set comparison algorithm is based on pair-wise comparisons of attributes with the following precedence:

- a) **priority1:** A user-configurable designation that a PTP Instance belongs to an ordered set of PTP Instances from which a Master PTP Instance is selected
- b) **clockClass:** An attribute defining the TAI traceability, synchronization state, and expected performance of the time or frequency distributed by a Boundary Clock or Ordinary Clock
- c) **clockAccuracy:** An attribute defining the accuracy of the Local PTP Clock of a PTP Instance
- d) **offsetScaledLogVariance:** An attribute defining the stability of the Local PTP Clock of a PTP Instance
- e) **priority2:** A user-configurable designation that provides finer grained ordering among otherwise equivalent PTP Instances
- f) **clockIdentity:** A tie-breaker based on unique identifiers for PTP Instances on different PTP Nodes

In addition to this precedence order, the “distance” measured by the number of Boundary Clocks between the local PTP Instance and the potential PTP Grandmaster Instance is used when two Announce messages reflect the same potential PTP Grandmaster Instance. The distance is indicated in the stepsRemoved field of Announce messages. This condition can occur in PTP Networks with cyclic paths not removed by a protocol outside of PTP. The data set comparison algorithm unambiguously selects one of the two PTP Instances as “better” or as “topologically better”.

The state decision algorithm determines whether the next state of the PTP Port—the recommended state—will be MASTER, SLAVE, or PASSIVE based on the results of the data set comparison algorithm and whether the PTP Instance’s clockClass is less than 128 (see 9.3.3). This recommended state is then evaluated by the PTP Port’s protocol engine based on the current state of the protocol state machine to determine the actual next PTP Port state.

6.6.2.4 Simple master–slave hierarchy

The process establishing a master–slave hierarchy among the Ordinary Clocks and Boundary Clocks in a domain is illustrated in Figure 13.

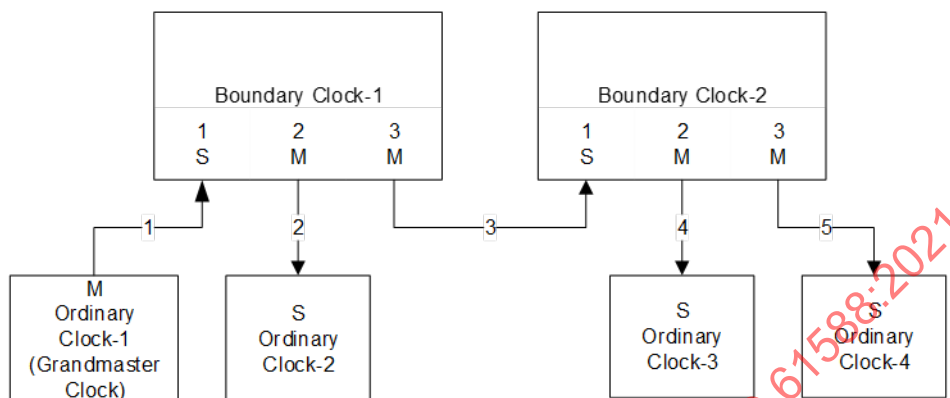


Figure 13—Simple master–slave PTP Instance hierarchy

In this example, Ordinary Clock-1 is at the root of the hierarchy and is called the “Grandmaster PTP Instance.” Port-1 of Boundary Clock-1 is in the SLAVE state (as indicated by the S) indicating that the Local PTP Clock of Boundary Clock-1 synchronizes to the Grandmaster Clock. All other PTP Ports on Boundary Clock-1 are in the MASTER state, and the Local PTP Clocks of the PTP Instances connected to them synchronize to the Local PTP Clock of Boundary Clock-1. Thus, Boundary Clock-2 is a Slave PTP Instance to Boundary Clock-1 and so forth. Only Ordinary Clocks and Boundary Clocks maintain this form of state, and only Boundary Clocks establish the branch points in the master–slave hierarchy (i.e., paths 1, 2, 3, 4, and 5 may contain Transparent Clocks, but these PTP Instances do not participate in the master–slave hierarchy and do not maintain this form of state).

6.6.2.5 Pruning mesh topologies

Figure 14 illustrates the case where a mesh PTP Network is reduced to a tree-structured master–slave hierarchy by the protocol. This occurs when the underlying bridging or routing protocols do not eliminate cyclic paths.

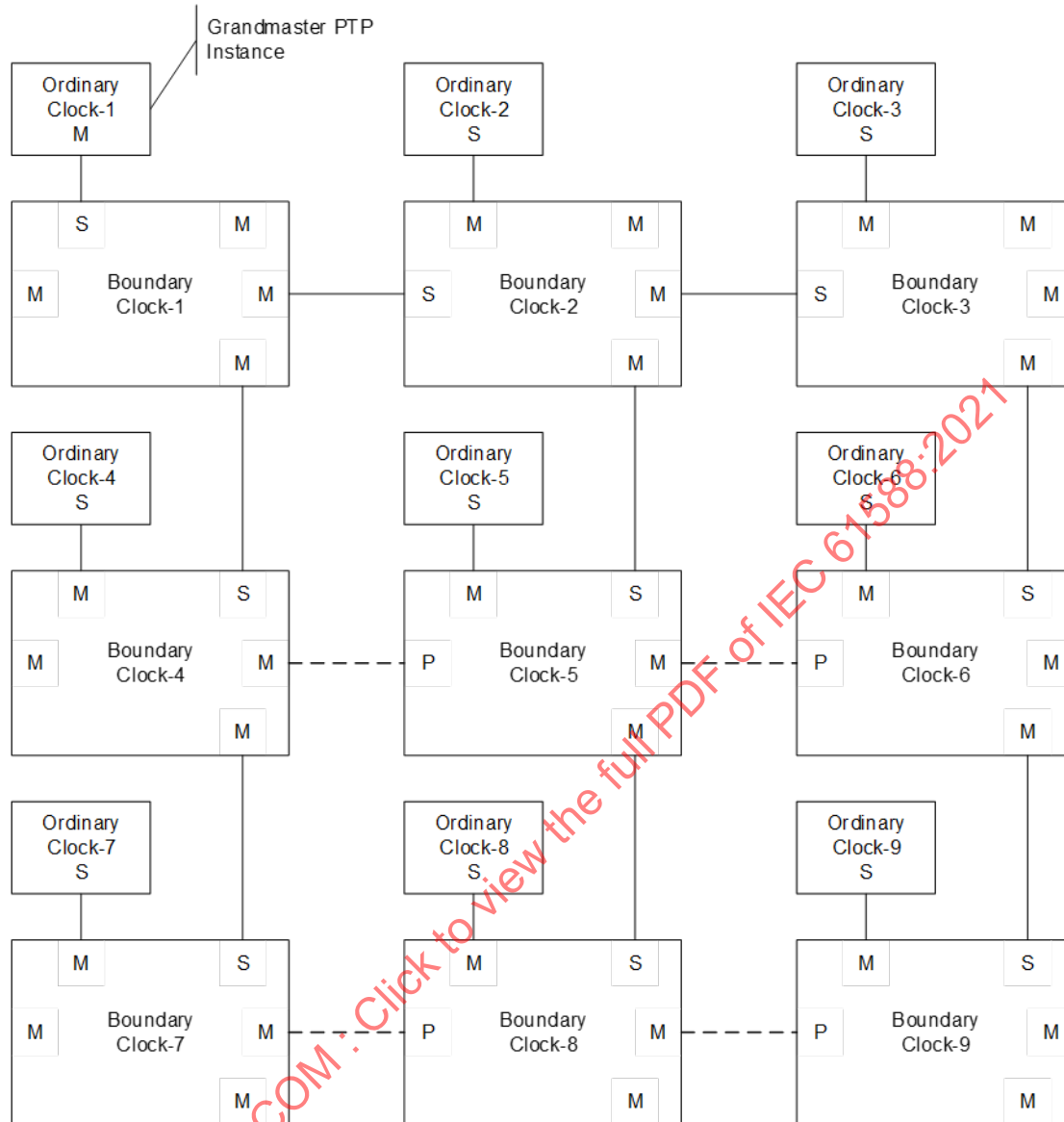


Figure 14—Pruned mesh topology

In Figure 14, Ordinary Clock-1 is assumed to have been selected as the Grandmaster PTP Instance by the best master clock algorithm. In the Boundary Clocks, the PTP Port states have been selected by the best master clock algorithm as shown to construct the tree. The pruned paths are shown in dashed lines. For each Boundary Clock, one PTP Port is selected by the best master clock algorithm as the PTP Port in the SLAVE state. The other PTP Ports are set to either the MASTER or the PASSIVE state. The best master clock algorithm ensures that a single PTP Port in the MASTER state is selected on each PTP Communication Path.

6.6.2.6 Segmentation of a domain by the BMCA

In a PTP Network containing two or more Ordinary Clocks or Boundary Clocks with characteristics that qualify them as potential Grandmaster PTP Instances, the BMCA will segment the PTP Network such that the Local PTP Clocks of the PTP Instances in each segment synchronize to the Grandmaster Clock in that segment. This is illustrated in Figure 15 where initially Ordinary Clock-A is the Grandmaster PTP Instance and there is a single domain.

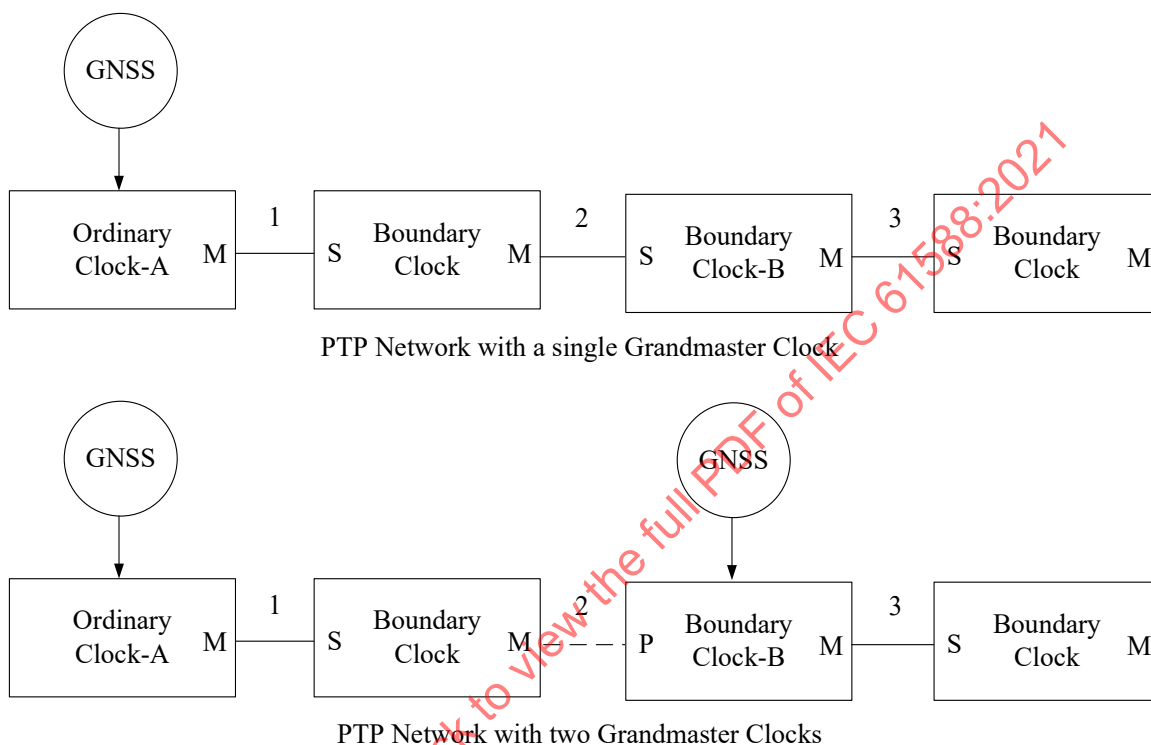


Figure 15—PTP Network split into two domains

When the clockClass of Boundary Clock-B upgrades, perhaps by obtaining time from a GNSS system, the PTP Network splits into two domains separated, for example, by the connection number 2 and with a PTP Port in the PASSIVE state on Boundary Clock-B. Boundary Clock-B is now the Grandmaster PTP Instance for the domain for PTP Instances on the right side of connection-2, while Ordinary Clock-A remains the Grandmaster PTP Instance for PTP Instances on the left side of connection-2.

NOTE—In Figure 15, since both GNSS sources are a primary reference time source (i.e., same time), and PTP Instance Time accuracy can degrade as it passes through a Boundary Clock, the segmentation represents the best hierarchy. This sort of segmentation occurs when clockClass of each PTP Instance is 127 or less. If this sort of segmentation is not desired for a given application, clockClass greater than 127 can be used.

6.6.3 Synchronizing Ordinary Clocks and Boundary Clocks

In a PTP Network, Ordinary Clocks or Boundary Clocks synchronize by exchanging PTP timing messages on the PTP Communication Path linking the two PTP Instances. For example, in Figure 13, Boundary Clock-1 synchronizes to Ordinary Clock-1 by exchanging PTP messages on PTP Communication Path 1.

The basic pattern of PTP timing message exchange is illustrated in Figure 16.

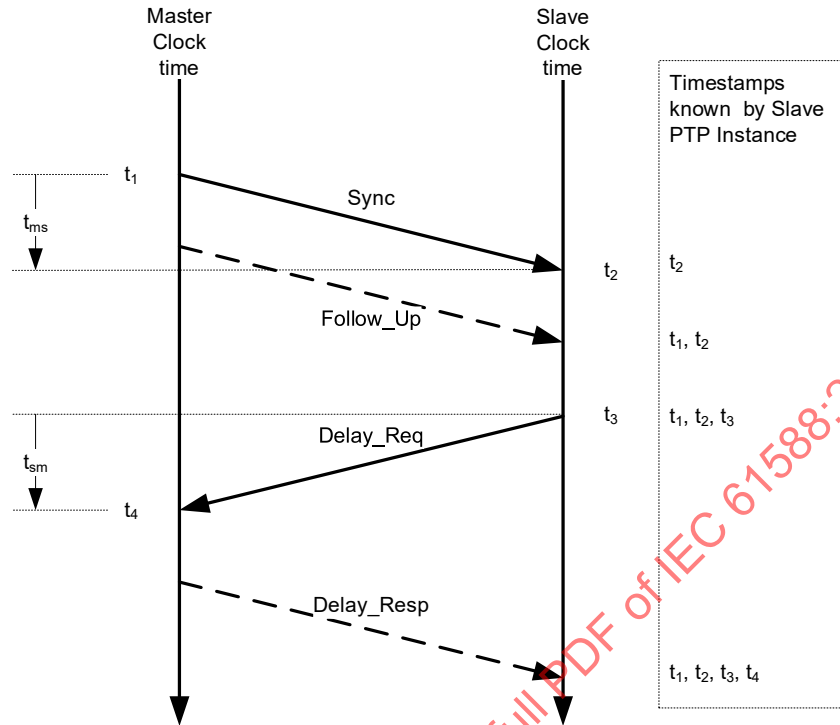


Figure 16—Basic PTP timing message exchange

The basic PTP timing message exchange pattern is as follows:

- The Master PTP Instance sends a Sync message to the Slave PTP Instance and notes the time t_1 at which it was sent.
- The Slave PTP Instance receives the Sync message and notes the time of reception t_2 .
- The Master PTP Instance conveys to the Slave PTP Instance the timestamp t_1 by:
 - Embedding the timestamp t_1 in the Sync message. This requires some sort of hardware processing for highest accuracy and precision, or
 - Embedding the timestamp t_1 in a Follow_Up message.
- The Slave PTP Instance sends a Delay_Req message to the Master PTP Instance and notes the time t_3 at which it was sent.
- The Master PTP Instance receives the Delay_Req message and notes the time of reception t_4 .
- The Master PTP Instance conveys to the Slave PTP Instance the timestamp t_4 by embedding it in a Delay_Resp message.

At the conclusion of this exchange of PTP messages, the Slave PTP Instance possesses all four timestamps. These timestamps may be used to compute the offset of the Slave Clock with respect to the Master Clock and the <mean propagation time>, that is, the path delay of PTP messages between the two clocks, which in Figure 16 is the mean of t_{ms} and t_{sm} .

The computation of offset and propagation time assumes that the master-to-slave and slave-to-master propagation times are equal. Any asymmetry in propagation time introduces an error in the computed value of the clock offset. The computed mean propagation time differs from the actual propagation times due to the asymmetry.

6.6.4 Measuring PTP Link propagation delay using peer-to-peer delay mechanism

The measurement of the PTP Link delay between two PTP Ports that implement the peer-to-peer delay mechanism (see 11.4) is illustrated in Figure 17. This measurement is conducted by all PTP Ports implementing the mechanism. Thus, both PTP Ports sharing a PTP Link independently make the measurement, and as a result, both PTP Ports know the PTP Link delay. This allows the corrections outlined in 6.5.4 to be made irrespective of the direction taken by a Sync message. It is important that this measurement occurs even on PTP Ports otherwise blocked by non-PTP algorithms used to eliminate cyclic topologies, so that up-to-date PTP Link delay measurements are available for all PTP Links in the event of a change in the path taken by a Sync message (see 6.5.4).

The PTP Link delay measurement starts with Port-1 issuing a Pdelay_Req message and generating a timestamp t_1 for the Pdelay_Req message. Port-2 receives the Pdelay_Req message and generates a timestamp t_2 for this PTP message. Port-2 issues a Pdelay_Resp message and generates a timestamp t_3 for this PTP message. To minimize errors due to any frequency offset between the two PTP Ports, Port-2 issues the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message.

Port-2 either:

- Conveys the difference between the timestamps t_2 and t_3 in the Pdelay_Resp message, or
- Conveys the difference between the timestamps t_2 and t_3 in a Pdelay_Resp_Follow_Up message, or
- Conveys the timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively.

Port-1 generates a timestamp t_4 upon receiving the Pdelay_Resp message. Port-1 then uses these four timestamps to compute the mean PTP Link delay.

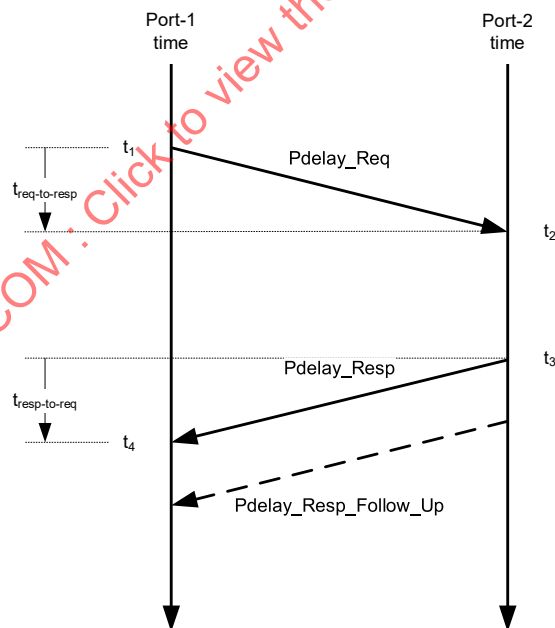


Figure 17—PTP Link delay measurement

Any asymmetry in the propagation times $t_{req-to-req}$ and $t_{resp-to-req}$ introduces an error into the computed value of the PTP Link delay. If the mechanisms producing the timestamps t_1 , t_2 , t_3 , and t_4 (i.e., the Timestamping Clocks) do not share the same definition of a second as the Grandmaster Clock, a possibly significant error is introduced in the PTP Link delay measurement. The source of this error is a small difference between the

frequency of the Timestamping Clocks in the Grandmaster Clock and peer-to-peer PTP Instances. In addition, if there is a difference between the frequency of the Timestamping Clocks in the two peer-to-peer PTP Instances, there is a small but possibly significant error due to the turnaround time, $t_3 - t_2$, in the peer-to-peer PTP Instance. If this error is significant, then the Timestamping Clock in the peer-to-peer PTP Instance can be frequency synchronized (i.e., syntonized) to that of the Timestamping Clock of the Grandmaster Clock, or the error can be computed and corrected based on the evaluated frequency difference between the two Timestamping Clocks.

6.6.5 Generation of PTP message timestamps

A timestamp is generated at the time of transmission and reception of any PTP event message. The timestamp generation occurs when the PTP message's timestamp point crosses the boundary between the PTP Instance and the network.

The generation of the timestamps, indicated in Figure 16 and Figure 17, is modeled in Figure 18.

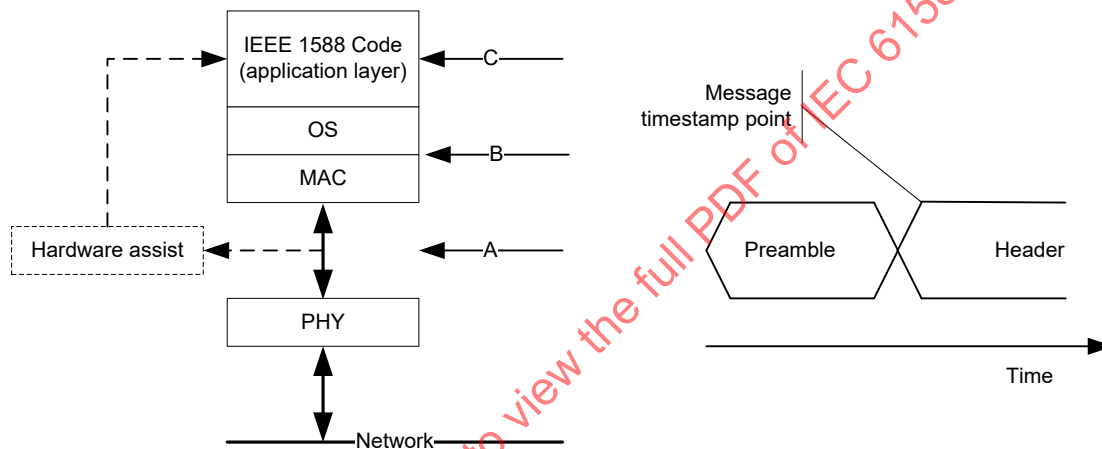


Figure 18—Timestamp generation model

PTP messages are issued from the PTP application code in one PTP Instance and received and processed by the PTP application code in another PTP Instance. These PTP messages typically have a preamble specified by the physical layer of the communication protocol in use on the PTP Network. The preamble is followed by one or more protocol specific headers and then user data such as the PTP payload. For every such transport mechanism, this standard specifies a particular point in PTP event messages, as a distinguished point termed the message timestamp point (see 7.3.4.1), which applies to many such transport mechanisms, including full-duplex Ethernet. As a PTP message traverses the protocol stack in a PTP Instance, the timestamps are generated when the message timestamp point passes a defined point in the stack. This point may be in the application layer, illustrated by “C” in Figure 18, in the kernel or interrupt service routines, illustrated by “B”, or in the physical layer of the protocol stack illustrated by “A”. In general, the closer this point is to the actual network connection, the smaller the timing errors introduced by fluctuations in the time taken to traverse the lower layers. In cases where timestamps are generated in the physical layer, some sort of hardware assist circuitry, illustrated by the dotted lines, is often employed. In this case, the timestamp is conveyed to the PTP code via a path outside of the normal path followed by the PTP event message itself. To ensure that the timestamps are associated with the correct PTP message, the hardware assist often captures additional information from the PTP event message that is passed with the timestamp to the PTP code.

Implementations that generate PTP timestamps at the physical layer need to take the mapping to the on-the-wire format (see 5.4) into account when designing packet recognizers, timestamp generators, and any other form of hardware assist to PTP that is done at the physical layer.

It is possible to design PTP Nodes where the timestamps are generated between the MAC layer and the PHY or even within the PHY. In such PTP Nodes, it is likely that all or part of the PTP timestamping will be embedded and executed within low-level silicon without the use of an operating system.

6.6.6 Potential errors in measuring path delays and residence time

6.6.6.1 Potential errors in measuring path delays and residence time due to differing rates of the Timestamping Clocks in PTP Instances in a domain

Errors in measuring path delays and residence time can occur when the PTP Instances involved in a particular measurement do not agree on the metric used in the measurements; that is, their Timestamping Clocks are not syntonized to the Grandmaster Clock of the domain. As an example, consider the peer-to-peer PTP Network of Figure 19.

In many implementations of Boundary Clocks and Ordinary Clocks, the Timestamping Clock is the Local PTP Clock that is synchronized by means of a servo loop, such that the measured offset of the Local PTP Clock, for example, of Boundary Clock BC-B, from the Local PTP Clock of the Grandmaster PTP Instance is minimized. This technique does require careful design of the servo loop, for example, by controlling gain peaking and overshoot. In Figure 19, assume that the servo loops in BC-B and Ordinary Clock OC-D have resulted in the frequency of the Local PTP Clocks in these PTP Instances agreeing with that of the Grandmaster Clock by 0.1 ppm and 0.2 ppm, respectively.

Next consider the situation in peer-to-peer Transparent Clock TC-C. Here the measurement of the residence time of PTP event messages is based on the Timestamping Clock that is the Local Clock. The Local Clock is commonly free running and, in this example, differs from the frequency of the Local PTP Clock of the Grandmaster PTP Instance by 100 ppm. If the residence time is about 10 ms, the measurement error introduced by the discrepancy in the frequencies is 1000 ns ($10^{-4} \times 10^{-2} \text{ s} = 10^{-6} \text{ s}$). This error would appear in the computation of the offset from master in OC-D.

Similar errors can be introduced in path delay measurements when the frequencies of the Timestamping Clocks involved do not agree with the frequency of the Grandmaster Clock of the domain. For example, the frequency differences between the Local PTP Clocks of BC-B and Grandmaster Clock GM-A, in this example, 0.1 ppm, will contribute path delay measurement errors of 0.3 ps for a path length of 1 km; that is, $10^{-7} \times 10^3 \text{ m} / (3 \times 10^8 \text{ m/s}) = 3.3 \times 10^{-13} \text{ s}$. Additional errors can occur due to turnaround time, that is, the time between the receipt of Pdelay_Req and sending of Pdelay_Resp, or the receipt of Sync and sending of Delay_Req.

Whether the errors in path delay and residence time are significant depends both on the sum of these errors and on the error budget of the application. For example, while a residence time error of 1000 ns from TC-C might be marginal in an application, the use of a 10 ppm oscillator as the Local Clock in TC-C reduces the potential error to 100 ns, which might provide sufficient margin of error.

If these syntonization discrepancy produced errors that exceed the error budget, correction methods must be invoked. These methods include:

- Using oscillators with improved frequency accuracy, for example, 0.1 ppm rather than 100 ppm.
- Synchronizing and syntonizing with servos as discussed in the example.
- Syntonizing the Timestamping Clocks in all PTP Instances to within the required tolerance with respect to the Grandmaster Clock, for example, the SyncE mechanism of ITU-T G.8262 [B34] or G.8273.2 [B36], or the high accuracy mechanism of this standard, see Annex L. This works for Boundary Clock, Ordinary Clocks and Transparent Clocks.

- Using the cumulative frequency transfer option of 16.10, which describes an alternate method of compensating for frequency discrepancies between the Timestamping Clocks (see 6.6.6.2). This works for Boundary Clock, Ordinary Clocks and Transparent Clocks.

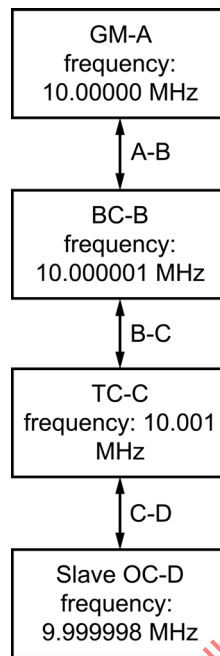


Figure 19—Simple peer-to-peer PTP Network

6.6.6.2 Correcting for differing Timestamping Clock rates—<neighborRateRatio>, <cumulativeRateRatio>

6.6.6.2.1 General

As noted in 6.6.6.1, errors can be introduced into the computation of <offsetFromMaster> (see 11.2), due to incorrect measurement of path delays and residence times, resulting from small differences in the rates of the Timestamping Clocks (i.e., Local Clocks or Local PTP Clocks) used for these measurements. For <neighborRateRatio>, these differences can be modelled as the ratio of the frequency of the Local Clock or Local PTP Clock of the upstream PTP Instance to the frequency of the Local Clock or Local PTP Clock of this PTP Instance. For <cumulativeRateRatio>, these differences can be modeled as the ratio of the frequency of the Grandmaster Clock to the frequency of the Local Clock or Local PTP Clock of this PTP Instance. The <neighborRateRatio> and the <cumulativeRateRatio> parameters defining these differences are discussed in the following examples.

The use of these parameters is discussed in the cumulative frequency transfer option of 16.10 and in the Common Mean Link Delay service of 16.6.

The nature of frequency-difference-induced errors can be seen from the following examples in 6.6.6.2.2 and 6.6.6.2.3.

6.6.6.2.2 Example using unsynchronized Local Clocks as the Timestamping Clock

Figure 20 illustrates a network containing two Grandmaster PTP Instances, GM-A and GM-B in domains A

and B, respectively. The two Slave PTP Instances, F and G, are also in domains A and B, respectively. These PTP Instances are connected by peer-to-peer Transparent Clocks C, D, and E.

The value of the frequency shown for each Timestamping Clock of a PTP Instance is the frequency of the oscillator that determines the rate at which time advances in the Local Clock of the PTP Instance. In this example, the oscillators are assumed to have frequency accuracy on the order of 100 ppm.

Suppose that Grandmaster PTP Instances GM-A and GM-B both simultaneously measured the time difference $\Delta E1-2$ between two events E1 and E2. If GM-A determined that $\Delta E1-2$ was 1 ms, GM-B would measure $\Delta E1-2$ to be 1.0002 ms since the Timestamping Clock of GM-B runs faster than that of GM-A by factor of 1.0002. Similarly, if $\Delta E1-2$ was measured using the Timestamping Clock of TC-E, the result would be 0.9996 ms since the Local Clock of TC-E runs slower than that of GM-A by a factor of 0.9996.

Suppose $\Delta E1-2$ is in fact the residence time for any Sync message traversing TC-E and is measured by the Timestamping Clock of TC-E to be 1 ms.

This means that the `<offsetFromMaster>` computed at Slave-F with respect to GM-A will include a TC-E residence time error of -0.0004 ms just due to the difference in the frequencies of TC-E and GM-A. Similarly, the error due to this residence time for the `<offsetFromMaster>` computed at Slave-G with respect to GM-B is -0.0006 ms.

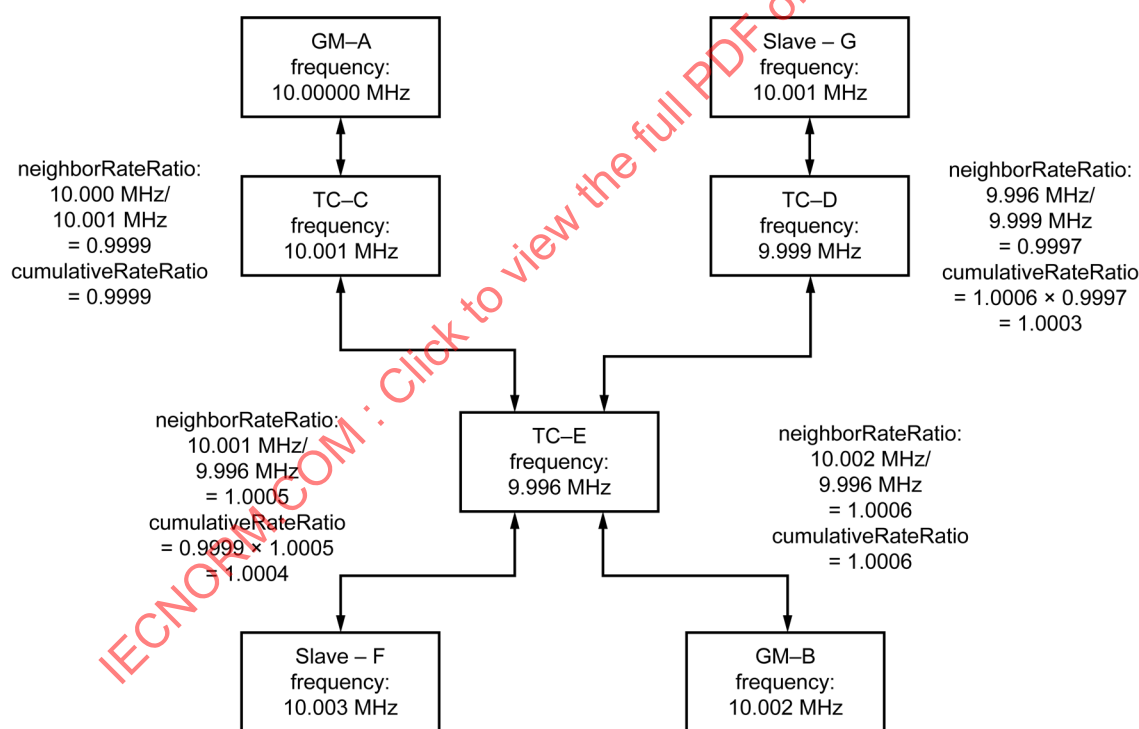


Figure 20—Example with unsynchronized Timestamping Clocks

Similar errors in the measurement of residence times and path delays occur at all intermediate PTP Instances where the frequency of the Local Clock differs from that of the Grandmaster Clock of the domain. The frequency deviations illustrated in Figure 20 are on the order of 100 ppm, a reasonable value for inexpensive oscillators. Whether the measurement errors due to these frequency differences are significant depends on the application.

Also, shown in the figure are computed values for the $\langle \text{neighborRateRatio} \rangle$ and values of $\langle \text{cumulativeRateRatio} \rangle$ at selected points in the PTP Network. For example, the $\langle \text{neighborRateRatio} \rangle$ at TC-C is the ratio of the frequency of the Timestamping Clock of the upstream PTP Instance, that is, GM-A, to that of the Timestamping Clock of TC-C. The $\langle \text{cumulativeRateRatio} \rangle$ is the product of neighborRateRatios between a PTP Instance and the Grandmaster PTP Instance of the domain. Thus, for example, the $\langle \text{cumulativeRateRatio} \rangle$ at TC-E in the domain of GM-A is 1.0004.

The $\langle \text{cumulativeRateRatio} \rangle$ can be used to correct the measured value of residence time to significantly reduce the errors discussed above. Again, using the numbers from the example, the residence time at TC-E based on the Timestamping Clock of TC-E is 1 ms. Multiplying this value by the $\langle \text{cumulativeRateRatio} \rangle$ of 1.0004 yields 1.0004 ms, which is the correct value when measured with the timescale of GM-A.

6.6.6.2.3 Example using syntonized Local Clocks as the Timestamping Clocks

Except for the Timestamping Clock in GM-A, which is presumably syntonized to its external source of time, (e.g., GPS), Figure 21 illustrates the same PTP Network but with all Local Clocks, and so Timestamping Clocks syntonized to the Local PTP Clock of GM-B. In this case, all $\langle \text{neighborRateRatio} \rangle$ and $\langle \text{cumulativeRateRatio} \rangle$ in the domain of GM-B will be 1.0000 and residence times and path delay measurements in this domain will not require correction.

NOTE 1—In reality, syntonization is not perfect. This means that $\langle \text{neighborRateRatio} \rangle$ and $\langle \text{cumulativeRateRatio} \rangle$ in the domain of GM-B will be very close to 1.0000 and residence times and path delay measurements in this domain might still require corrections depending on the deviation of the ratios from 1.0000 and the application requirements.

NOTE 2—In the case where the Local Clocks are syntonized to the Grandmaster Clock of one of the domains, the frequency of each syntonized Local Clock is equal to the frequency of the Local PTP Clock for that domain.

NOTE 3—Syntonized Local Clocks are commonly used in telecommunications applications among others. However, in the domain of GM-A, the $\langle \text{cumulativeRateRatio} \rangle$ will all be 0.9998 due to the frequency difference between GM-A and GM-B. As before, the $\langle \text{cumulativeRateRatio} \rangle$ can be used to correct measurements of residence time and path delay into the timescale of GM-A.

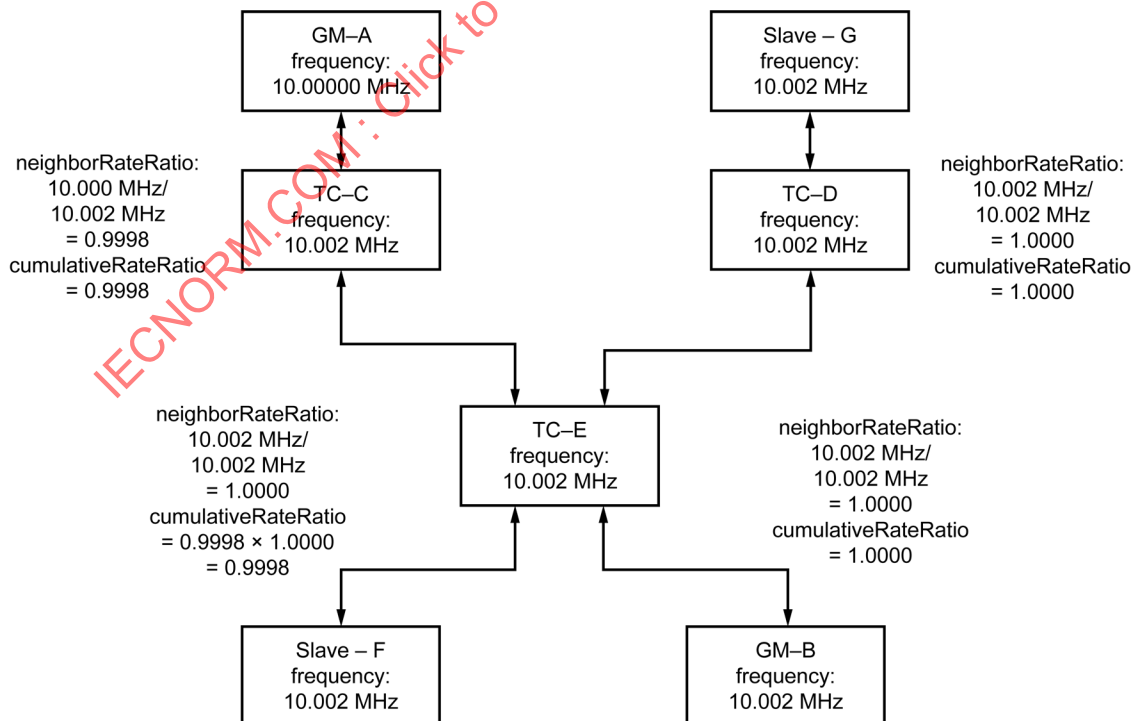


Figure 21—Example with syntonized Local Clocks

6.7 PTP communications overview

6.7.1 PTP communication topology

6.7.1.1 General

In PTP Networks composed solely of Boundary Clocks and Ordinary Clocks, the operation of PTP produces an acyclic graph structure for PTP messages irrespective of the actual underlying network connectivity, following techniques described by Perlman [B45].

In all other cases, PTP assumes that the underlying bridging or routing protocols ensure that PTP message forwarding avoids loops. In particular, the protocol assumes that multicast PTP messages are not looped indefinitely within the PTP Communication Path. The protocol does not assume that a multicast PTP message sent by one PTP Port is received on only one of the recipient Boundary Clock's PTP Ports. The protocol does not assume that only a single copy of a multicast PTP message sent by one PTP Port is received on another PTP Port; however, receipt of duplicate copies might impact the precision of time transfer, and therefore, networks should be designed to avoid this behavior.

6.7.1.2 Hierarchical topology

Different applications favor different topologies. For many PTP Networks, the hierarchical topology illustrated in Figure 22 is preferred.

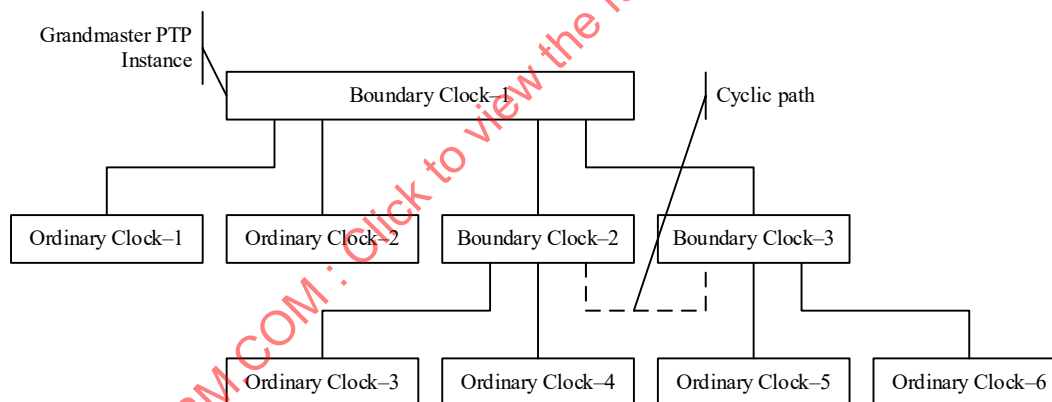


Figure 22—Hierarchical topology

With the exception of the “cyclic” path shown in dashed lines, the other PTP Instances in Figure 22 form a tree-structured hierarchy with the Boundary Clocks forming the branch points in the tree. Since Boundary Clocks can have many PTP Ports, this topology allows the Local PTP Clocks of a large number of PTP Instances to be synchronized with only a few Boundary Clocks between any Slave PTP Instance and the Grandmaster PTP Instance. In this example, assume that the dashed path is not present. If Boundary Clock-1 is selected as the Grandmaster PTP Instance as shown, the maximum number of intervening Boundary Clocks is 1. The worst case would be if one of the Ordinary Clocks, say, Ordinary Clock-3, is selected as the Grandmaster PTP Instance. Then the maximum number of Boundary Clocks to the most distant Slave PTP Instance, say, Ordinary Clock-6, would be 3.

As noted in 6.2, the protocol expects the underlying topology to avoid forwarding loops. In Figure 22, if the dashed path is present, a loop exists involving three Boundary Clocks. The operation of the best master clock and state decision algorithms noted in 6.6.2 breaks this loop for PTP messages.

6.7.1.3 Linear topology

Some applications require long linear topologies as shown in Figure 23, rather than the star or hierarchical topologies of Figure 22. Figure 23 shows two long linear chains of end-to-end Transparent Clocks with Boundary Clock-1 as the Grandmaster PTP Instance.

Synchronization between an Ordinary Clock and a Boundary Clock involves an exchange of PTP timing messages between the two PTP Instances, for example, over path-A. Based on the PTP timing messages, the Slave PTP Instance of such a master–slave pair executes some sort of servo mechanism to reduce the clock offset errors.

End-to-end Transparent Clocks measure the time spent by a PTP event message within the Transparent Clock and retransmit PTP timing messages as specified in 6.5.3. These “residence” times are accumulated in the correctionField in the PTP timing messages, which allows the Slave PTP Instance to correct the timestamps, effectively removing the timing fluctuations that would otherwise be introduced. The penalty is that the Master PTP Instance, in this example, Boundary Clock-1, processes PTP timing messages (i.e., Delay_Req) from all Slave PTP Instances in the linear chain, rather than just from an adjacent Boundary Clock.

NOTE—This scaling penalty on the Master PTP Instance can be avoided by using Boundary Clocks or peer-to-peer Transparent Clocks instead of end-to-end Transparent Clocks.

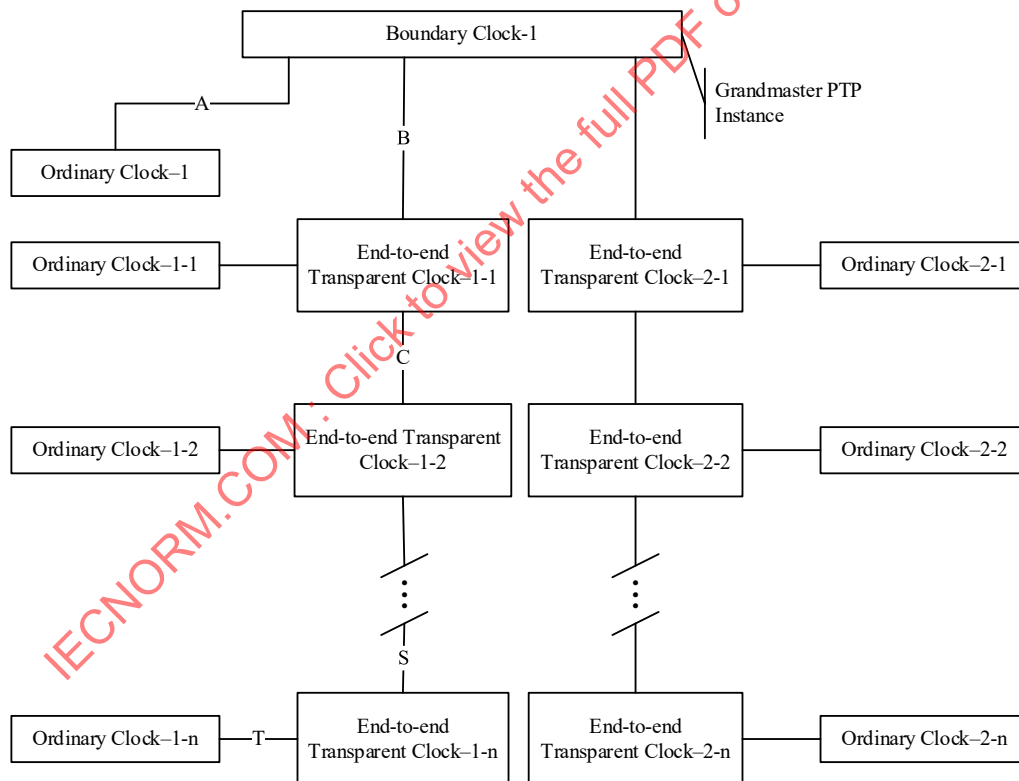


Figure 23—Linear topology

6.7.1.4 Rapid reconfiguration in multiply connected topologies

In many applications, PTP Instances are placed in a multiply connected topology, for example, a mesh, as illustrated in Figure 24 or a ring in which alternate paths are logically removed to produce an acyclic

topology by a protocol outside of PTP. In the event of a path failure, these external protocols reconfigure the network to restore connectivity. Since PTP operates on top of this underlying, rapidly reconfiguring network, PTP might have to readjust the corrections for path delay between Master PTP Instance and Slave PTP Instance after a reconfiguration.

The peer-to-peer Transparent Clock can be suitable for use in this environment.

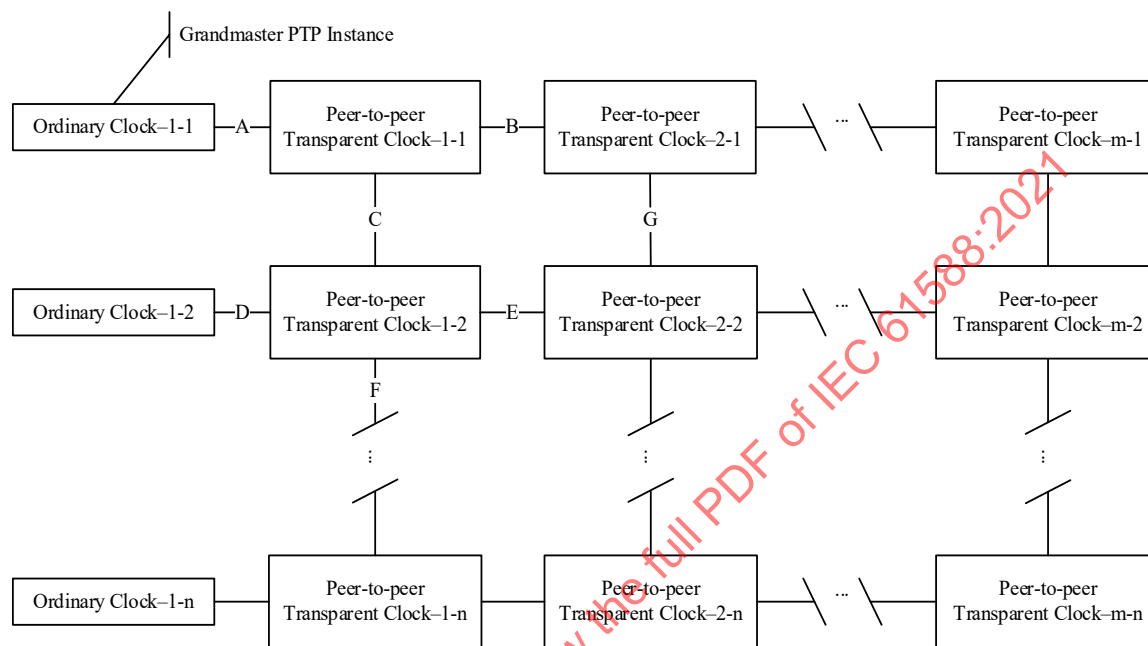


Figure 24—Multiply connected topology

Shown in Figure 24 are several peer-to-peer Transparent Clocks connected in a mesh topology. The operation of the peer-to-peer Transparent Clock depends on some external protocol eliminating cyclic paths in the network. As in the case of end-to-end Transparent Clocks, the peer-to-peer Transparent Clocks are usually associated with an Ordinary Clock for use in a sensor or other device. The difference between the two types of PTP Instances is in the way that the path delay corrections are made.

Suppose that initially the path between Ordinary Clock-1-1 (the Grandmaster PTP Instance) and Ordinary Clock-1-2 (the Slave PTP Instance) was A, B, G, E, D as determined by some non-PTP protocol. A Sync message from Ordinary Clock-1-1 would be corrected by peer-to-peer Transparent Clock-1-1 for residence time in the peer-to-peer Transparent Clock and for PTP Link delay A. Likewise peer-to-peer Transparent Clock-2-1 would further correct for its residence time and PTP Link delay B, and so on, so that Ordinary Clock-1-2 receives a Sync message corrected for residence times in peer-to-peer Transparent Clocks 1-1, 2-1, 2-2, and 1-2 and PTP Link delays A, B, G, and E. Ordinary Clock-1-2, which also supports the peer-to-peer delay mechanism, corrects for the last PTP Link delay D.

Assume that the network is reconfigured such that the new path between these two clocks is now A, C, D. Peer-to-peer Transparent Clock-1-1 does the same correction as before. However, in this case, peer-to-peer Transparent Clock-1-2 now receives the Sync message directly from peer-to-peer Transparent Clock-1-1 and, therefore, corrects for its residence time and PTP Link delay C, which it has previously measured. It is this prior measurement of PTP Link delays on all PTP Links, whether active or not, that allows this rapid reconfiguration.

6.7.1.5 Bridging between different network protocols

There is no requirement that all of the PTP Communication Paths use the same underlying communication media or technology. Boundary Clocks can be used to bridge between different network transport technologies, as illustrated in Figure 25.

In Figure 25, Boundary Clock-1 and Ordinary Clock-1, Ordinary Clock-3, and Ordinary Clock-5 are assumed to communicate via network paths labeled A implementing one technology, for example, UDP/IP. Boundary Clock-2 and Ordinary Clock-2, Ordinary Clock-4, and Ordinary Clock-6 communicate via network paths labeled B implementing a second technology, for example, DeviceNet. One of the Boundary Clocks, Boundary Clock-2 in Figure 25, is a bridge that supports technology A on PTP Port-2 and technology B on the remaining PTP Ports.

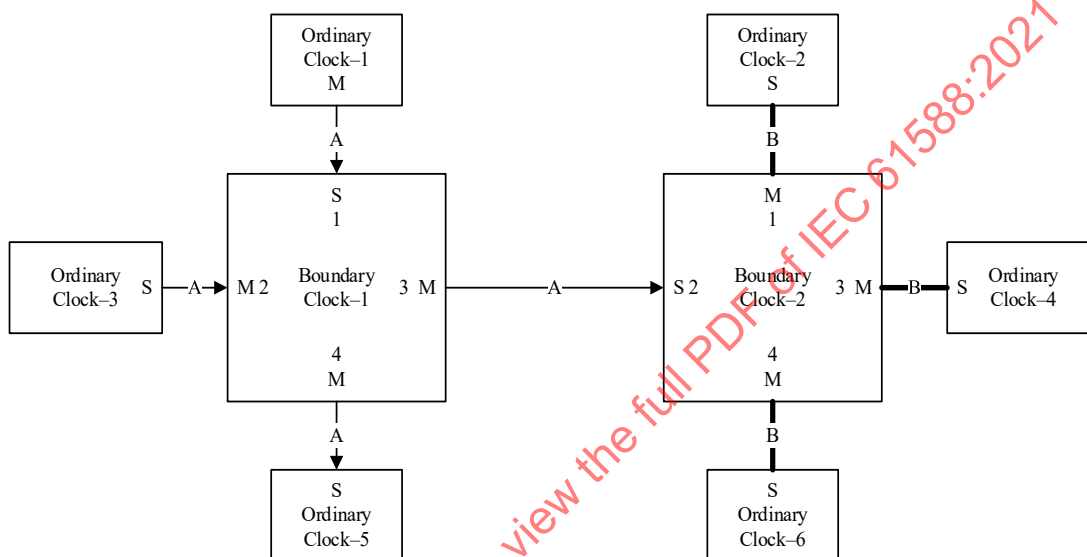


Figure 25—Bridging disparate technologies

In some cases, such bridging involves not only a change in network transport protocols, as in the above example between UDP/IP and DeviceNet, but also possibly other PTP characteristics such as update rates. In this case, the Boundary Clock is the only PTP Instance that maintains sufficient PTP state to perform the bridging function. In some cases, the only bridging function required is translating packet formats or other physical layer issues. In these cases, it is possible to design Transparent Clocks to perform the bridging since no PTP state information is required.

6.7.2 PTP Network startup

To provide more orderly behavior when a PTP Instance comes online, an Ordinary Clock or Boundary Clock listens for Announce messages from a Master PTP Instance for a configurable time interval. If no Announce message is received within this time, the PTP Instance assumes it is the Master PTP Instance until a “better” PTP Instance appears.

An additional mechanism to support more orderly reconfiguration of PTP Networks when PTP Instances are added or deleted, PTP Instance characteristics change, or connection topology changes is embodied in the PRE_MASTER state. In this state, a PTP Port behaves exactly as it would if it were in the MASTER state except that it does not place certain classes of PTP messages on the PTP Communication Path associated with the PTP Port. A PTP Port remains in this PRE_MASTER state long enough to allow changes to propagate from points in the PTP Network between the PTP Instance and possible Master PTP Instances visible from the PTP Port.

7. Characterization of PTP entities

7.1 Domains

7.1.1 General

A domain consists of one or more PTP Instances communicating with each other as defined by the protocol. A domain shall define the scope of PTP message communication, state, operations, data sets, and timescale.

Within a PTP Network, a domain is identified by two attributes: domainNumber and sdoId.

The domainNumber of a domain is an integer in the closed range 0 to 255.

The sdoId of a domain is a 12-bit integer in the closed range 0 to 4095. An sdoId is structured as a two-part attribute as follows:

- The most significant 4 bits are named the majorSdoId,
- The least significant 8 bits are named the minorSdoId.

The values of both the domainNumber and the sdoId for a domain appear in the common header of all PTP messages (see 13.3.1).

NOTE 1—In the 2008 edition, the attribute majorSdoId was named transportSpecific and its values were specified-by-design as either 0 or 1 in C.4, D.4, E.4, F.4, and G.4 of IEEE Std 1588-2008. The attribute minorSdoId did not exist in the 2008 edition, but its location in the common header was a reserved field, and therefore, its value for PTP messages transmitted by a 2008 PTP device are specified-by-design to be 0. See 7.1.4 for information about assigning values for sdoId specified in this edition of the standard.

NOTE 2—See Clause 18 for interactions between PTP Instances in different domains.

NOTE 3—The mechanism for limiting PTP operation and communications to a domain might involve, for example, communication-specific techniques such as router table configuration, limiting the physical connectivity, and measures of this clause and 16.5.

NOTE 4—Implementers of PTP Nodes need to consider the resources required to support multiple domains. This is particularly true for Ordinary Clocks and Boundary Clocks, which maintain much more state information than Transparent Clocks. The inability to process the protocol in a timely fashion due to resource limitations can lead to deterioration in the synchronization performance, thrashing, or failure of the protocol.

7.1.2 Domain identification semantics when not using the isolation option of 16.5

7.1.2.1 In Boundary and Ordinary Clocks

A PTP message received by a PTP Instance shall be considered for further processing by the protocol if and only if:

- a) The value of the domainNumber field of the common header is equal to the value of the defaultDS.domainNumber, and
- b) The value of the majorSdoId field of the common header is equal to the value of the majorSdoId portion, that is, the most significant 4 bits, of the sdoId (see 7.1.4).

The implementation of the comparison of majorSdoId values in implementations based on layer 2 IEEE 802.1 communications shall be in accordance with the specification in E.4.

NOTE—Backward compatibility with 2008 PTP devices is achieved as follows. A conformant 2008 PTP device receiving a layer-2 PTP message with an sdoId value greater than or equal to 200_{16} , for example, 303_{16} with any domainNumber will interpret the majorSdoId value, that is, 3_{16} , as the transport specific value and will reject the message per 2008 F.4. A conformant 2008 PTP device receiving a layer-3 PTP message with an sdoId value greater than 100_{16} , for example, 303_{16} with a domainNumber in the range 128 to 239 per Table 2 of this edition will reject the message based on domainNumber since domainNumbers in this range are reserved in the 2008 edition and will therefore not be used by a 2008 PTP device. In this case, the way the 2008 PTP device interprets the value of the sdoId is not relevant.

7.1.2.2 In Transparent Clocks

The domain attributes domainNumber and sdoId need to match for PTP messages that are associated with each other in protocol operation. The specifications for correctly associating the message pairs Sync and Follow_Up, Delay_Req and Delay_Resp, and Pdelay_Req Pdelay_Resp and Pdelay_Resp_Follow_Up for PTP protocol operations are in 10.2.1 and 10.3.1.

NOTE—There are cases where domainNumber and sdoId need not be tested because the received PTP message can be treated as a singleton and not as a member of a pair. For example, if the Transparent Clock can make residence time corrections to Sync messages, then the correction for residence time can be made on the correctionField of the Sync message, irrespective of whether there is or is not a Follow_Up message, and therefore, there is no need to check the association. This sort of example might not apply in the presence of additional requirements, (e.g., security) imposed in a profile or by the standard.

7.1.3 Domain identification semantics when the isolation option of 16.5 is being used

NOTE—In the 2008 edition, octet 5 of the common header is reserved and therefore has the value 0. In this edition, octet 5, the minorSdoId, still has the value 0 for majorSdoId values of 0 and 1. Therefore implementations can simply check all three fields, domainNumber, majorSdoId, and minorSdoId irrespective of whether the provisions of 16.5 are used or whether 2008 PTP devices are present in the PTP Network.

7.1.3.1 In Boundary and Ordinary Clocks

A PTP message received by a PTP Instance shall be considered for further processing by the protocol if and only if:

- The value of the domainNumber field of the common header is equal to the value of the defaultDS.domainNumber, and
- The value of the majorSdoId field of the common header is equal to the value of the majorSdoId portion, i.e. of the most significant 4 bits, of the defaultDS.sdoId member, and
- The value of the minorSdoId field of the common header is equal to the value of the minorSdoId portion, i.e. the least significant 8 bits, of the defaultDS.sdoId member.

The implementation of the comparison of majorSdoId values in implementations based on layer 2 IEEE 802.1 communications shall be in accordance with the specification in E.4.

NOTE—In the 2008 edition in Annex F (specifications for IEEE Std 802.3/Ethernet), the test on the match for majorSdoId (this attribute was named transportSpecific in the 2008 edition) was based on the Ethernet subtype, thereby permitting this match test to be easily implemented in hardware. This edition retains this specification but broadens the test to all transports. Matching tests for domainNumber, majorSdoId, and minorSdoId will typically be done in software for layer 3 transports.

7.1.3.2 In Transparent Clocks

The domain attributes domainNumber and sdoId must match for PTP messages that are associated with each other in protocol operation. The specifications for correctly associating the message pairs Sync and Follow_Up, Delay_Req and Delay_Resp, and Pdelay_Req Pdelay_Resp and Pdelay_Resp_Follow_Up for PTP protocol operations are in 10.2.1 and 10.3.1.

NOTE—There are cases where domainNumber and sdoId need not be tested because the received PTP message can be treated as a singleton and not as a member of a pair. For example, if the Transparent Clock can make residence time corrections to Sync messages, then the correction for residence time can be made on the correctionField of the Sync message, irrespective of whether there is or is not a Follow_Up message, and therefore there is no need to check the association. This sort of example might not apply in the presence of additional requirements, (e.g., security) imposed in a profile or by the standard.

7.1.4 Assigning values for the domainNumber and sdoId for a domain

The value of the domainNumber shall be configurable to values permitted in Table 2 unless the allowed values are further restricted by the applicable PTP Profile. The domainNumber is the primary mechanism for end users and system integrators to isolate the operation of a PTP Instance from PTP messages used in other domains. Standards organizations writing profiles may further restrict, but shall not expand, the range of domain numbers in Table 2. The value of the domainNumber attribute is a field of the common header (see 13.3).

The sdoId attribute is the primary mechanism for providing isolation of PTP Instances of Profiles specified by a QSDO from other PTP Instances operating under a PTP Profile specified by a different QSDO (see 16.5). The value of the sdoId attribute is represented in the common header (see 13.3).

The value of the sdoId attribute of a PTP Instance shall be the value of defaultDS.sdoId. The value of the sdoId attribute of a PTP Instance:

- a) May be specified-by-design if the value is either 000₁₆ or 100₁₆, else
- b) Except for the exception specified in C.4.2, shall be a value selected from the permitted sdoId values of Table 2. It may be specified-by-design or configurable.

The value of the domainNumber attribute applicable to a PTP Instance shall be a value from the range of values permitted for the selected sdoId value as specified in Table 2.

Values of combinations of sdoId and domainNumber that are not listed in Table 2 are reserved (4.2.7), and they shall not be used.

NOTE 1—sdoId values in the ranges 001₁₆ through 0FF₁₆ and 101₁₆ through 1FF₁₆ are not permitted in order to preserve backward compatibility.

Table 2—Permitted values of the domainNumber and sdold attributes

sdold	domainNumber values when PTP communications are based on Layer 2 IEEE Std 802, Annex F	domainNumber values when PTP communications are not based on Layer 2 IEEE Std 802	Permission to use the sdold granted to:
000 ₁₆	0–127 are permitted 128–255 are reserved	0–127 are permitted 128–255 are reserved	Implementations based on IEEE Std 1588-2008 and any implementations based on this edition where the expanded profile isolation capabilities of option 16.5 are not required
100 ₁₆	0–239 are permitted 240–255 are reserved	128–239 are permitted 0–127 and 240–255 are reserved	Implementations based on PTP Profiles developed by the IEEE 802.1 Working Group ^a
200 ₁₆	Per 16.6.3 240–255 are reserved	Per 16.6.3	IEEE 1588 Common Mean Link Delay Service; see 16.6
201 ₁₆ – 2FF ₁₆	Reserved	Reserved	IEEE 1588 Working Group ^b
300 ₁₆ – FFC ₁₆	0–239 are permitted 240–255 are reserved	128–239 are permitted 0–127 and 240–255 are reserved	Implementations based on Profiles specified by QSDOs using an sdold value in this range, obtained from the IEEE Registration Authority, to create PTP Profiles using option 16.5
FFD ₁₆ , FFE ₁₆	0–239 are permitted 240–255 are reserved	128–239 are permitted 0–127 and 240–255 are reserved	Implementations on a temporary basis only; for experimental implementations, see 4.2.9
FFF ₁₆	Reserved	Reserved	IEEE 1588 Working Group

^a IEEE 802.1 Working Group (<https://1.ieee802.org/>).^b IEEE Precise Networked Clock Synchronization Working Group. Also referred to in this document as the IEEE 1588 Working Group (<https://standards.ieee.org/project/1588.html>).

NOTE 2—For the transport of PTP over IEC 61588 type 10 (see Annex H), the mapping of majorSdold and minorSdold onto the transport layer is not defined (see H.4). Therefore, when Annex H is used, the application of sdold values other than 000₁₆ is outside the scope of this standard.

7.2 Timescales used in PTP

7.2.1 General

The timescale for a domain is established by the Grandmaster Clock.

The following two types of timescales are supported by PTP:

- **The timescale PTP:** This is the elapsed time since the PTP epoch measured using the second defined by International Atomic Time (TAI) (see NIST SP 330:2008 [B42], with the further amplification of *Proceedings of the 21st General Assembly of the IAU* [B48] for the definition of TAI, and Petit and Luzum [B47] for more information on TAI).
- **The timescale ARB:** In normal operation, the epoch is set by an administrative procedure. The epoch is permitted to be reset during normal operation. Between invocations of the administrative procedure, the timescale is continuous. Additional invocations of the administrative procedure may introduce discontinuities in the overall timescale. The unit of measure of time is determined by the Grandmaster Clock. The second used in the operation of the protocol may differ from the SI second.

NOTE— In the 2008 edition of this standard, 7.2 was named “PTP timescale”. It defined “timescale PTP” and “timescale ARB”. Ambiguously, throughout the 2008 edition, the term “PTP timescale” was used to mainly refer to “timescale PTP” and the term “ARB timescale” was used to refer to “timescale ARB”. In this edition of the standard, the terms “PTP timescale” and “ARB timescale” are not used while the terms “timescale PTP” and “timescale ARB” are used consistently. The name `ptpTimescale` of an attribute, `flagField` and data set member of the `timePropertiesDS` is retained in this edition of the standard.

7.2.2 Oscillator frequency

Oscillators used to establish or maintain the timescale of a PTP Instance (see 7.2.1) shall maintain a period, that is, the inverse of the frequency, deviating no more than 0.01% from the period of a perfect oscillator of the same nominal frequency.

7.2.3 Epoch

The epoch is the origin of the timescale of a domain.

The PTP epoch (the epoch of timescale PTP) is 1 January 1970 00:00:00 TAI.

NOTE 1—The common POSIX algorithms can be used for converting elapsed seconds since the PTP epoch to the ISO 8601:2004 printed representation of time of day on the TAI timescale; see ISO/IEC 9945:2003 [B30] and ISO 8601:2004 [B29].

NOTE 2—See Annex B for information on converting between common timescales.

7.2.4 UTC Offset

When the timescale is PTP, it is possible to calculate UTC time using the value of `<dLS>`, which is maintained in the `timePropertiesDS.currentUtcOffset` data set member (see 8.2.4.2). `<dLS>` is given by:

$$\text{<dLS>} = \text{TAI} - \text{UTC}$$

The value `<dLS>` is the offset between TAI and UTC. The value of `<dLS>` is derived from UTC-TAI (the negative of `<dLS>`) specified in IERS Bulletin C.

NOTE 1—As of 0 hours 1 January 2017 UTC, UTC was behind TAI by 37 seconds. At that moment, the PTP-defined value of `<dLS>` was +37 seconds, as designated in the applicable Bulletin C (see Clause 2).

NOTE 2—Leap-second events and the value of UTC-TAI are posted well in advance in IERS Bulletin C. A list of all leap-second events is maintained by the U.S. Naval Observatory [B54]. See also the USNO Time Service Department¹¹ for an extensive discussion of timescales, leap seconds, and related time issues.

NOTE 3—The value of `<dLS>` represents the difference between TAI and UTC. Since 1972, only integral values are permitted for this difference. Due to leap seconds, UTC cannot be correctly represented as a single integer but can be expressed in the ISO 8601:2004 print form [B29]. See also B.2.1 for an example of the use of the POSIX algorithm to compute the correct print form.

There are two mechanisms specified in this standard that enable computing UTC time from PTP Instance Time: the mechanism of this clause and the mechanism of 16.3. Unless otherwise specified in the applicable PTP Profile or other applicable standards documents, PTP Instances that compute UTC time shall use the mechanism of this clause to compute UTC from PTP Time.

¹¹ USNO Time Service Department (<http://tycho.usno.navy.mil/>).

NOTE 4—In IEEE Std 1588-2008, 16.3.1 specified that the mechanism of the Alternate Timescales, 16.3, not be used to distribute and compute UTC time. This is overly restrictive. The above specification is introduced to avoid interoperability issues.

When the mechanism of this subclause is selected, the values of `timePropertiesDS.leap59`, `timePropertiesDS.leap61`, and `timePropertiesDS.currentUtcOffset` shall be used when an internal representation of UTC based on the PTP Instance Time is computed. A discussion of the computational details is in B.2.2.

NOTE 5—See 16.3 for specifications on an alternate method of computing UTC time and for distributing offsets to compute time in other timescales.

For calculating time in other timescales in cases where the mechanism of 16.3 is not used, the PTP Instance shall make offset corrections to internal representations of other timescales using any localization or application-specific information maintained by the PTP Instance, for example, the time zone in the case of computing Eastern Standard Time from UTC.

When the timescale is ARB, the value of `<dLS>` shall not be used to compute UTC. In this case, the mechanisms specified in 16.3 may be used to distribute information enabling computation of UTC from the timescale ARB.

The mechanism for computing UTC or any other time shall not change the PTP Instance Time of a PTP Instance.

7.2.5 Measurement of time within a domain

Within a domain, time shall be measured as elapsed time since the epoch.

7.3 PTP communications

7.3.1 Messaging model

All messages, including PTP messages, shall be transmitted and received in conformance with the standards governing the transport, network, and physical layers of the communication paths used.

NOTE—As an example, consider IEEE 1588 PTP Instances, specifically including Transparent Clocks, running on IEEE 802.1Q communication paths. Suppose we have two Boundary Clocks separated by a Transparent Clock. The Transparent Clock entity (the PTP stack running above the MAC layer) is required to insert the appropriate MAC address of the Transparent Clock into the `sourceAddress` field of the Ethernet header for ALL messages it transmits. Other communication protocols can have similar requirements.

Although the standard is written based on the multicast model, an implementation may be created based on a unicast model, or on a mixed multicast/unicast model, provided that the behavior of the protocol is preserved.

Unless otherwise specified in this standard, the specifications for unicast communication are the same as the specifications for multicast communication. PTP Nodes that operate as bridges or routers, and that receive unicast PTP messages that have a destination address of a different PTP Node, shall forward the PTP messages according to the rules of the transport protocols.

7.3.2 PTP Message attributes

All PTP-related communications occur via PTP messages. PTP messages have the following attributes:

- Message class
- Message sourcePortIdentity
- Message type
- Message sequenceId
- Flags defining options

7.3.3 Message class

7.3.3.1 PTP event messages

The PTP event message class consists of the following PTP message types:

- a) **Sync:** A Sync message is transmitted by a Master PTP Instance to its Slave PTP Instances. It either contains the time of its transmission or is followed by a Follow_Up message containing this time. It may be used by a receiving PTP Instance to measure the packet transmission delay from the Master PTP Instance to the Slave PTP Instance. The appearance of a Sync message at the reference plane (see Figure 26) of a PTP Port is an event to which a PTP Instance shall assign a timestamp, the <syncEventIngressTimestamp> or <syncEventEgressTimestamp>, based on the value of the Timestamping Clock.
- b) **Delay_Req:** A Delay_Req message is a request for the receiving PTP Instance to use a Delay_Resp message to return the time at which the Delay_Req message was received. The appearance of a Delay_Req message at the reference plane of a PTP Port is an event to which a PTP Instance shall assign a timestamp, the <delayReqEventIngressTimestamp> or <delayReqEventEgressTimestamp>, based on the value of the Timestamping Clock.
- c) **Pdelay_Req:** A Pdelay_Req message is transmitted by a PTP Port to another PTP Port as part of the peer-to-peer delay mechanism (see 11.4) to determine the delay on the PTP Link between them.

The appearance of a Pdelay_Req message at the reference plane of a PTP Port is an event to which a PTP Instance shall assign a timestamp, the <pdelayReqEventIngressTimestamp> or <pdelayReqEventEgressTimestamp>, based on the value of the Timestamping Clock.

- d) **Pdelay_Resp:** A Pdelay_Resp message is transmitted by a PTP Port in response to the receipt of a Pdelay_Req message. There are several options for conveying timestamp information in the Pdelay_Resp message. They are as follows:
 - 1) The difference between the time of transmission of the Pdelay_Resp message and the time of receipt of the corresponding Pdelay_Req message is conveyed in the Pdelay_Resp message.
 - 2) The difference between the time of transmission of the Pdelay_Resp message and the time of receipt of the corresponding Pdelay_Req message is conveyed in a Pdelay_Resp_Follow_Up message that follows the Pdelay_Resp message.
 - 3) The time of receipt of the corresponding Pdelay_Req message is conveyed in the Pdelay_Resp message, and the time of transmission of the Pdelay_Resp message is conveyed in a Pdelay_Resp_Follow_Up message that follows the Pdelay_Resp message.

The appearance of a Pdelay_Resp message at the reference plane of a PTP Port is an event to which a PTP Instance shall assign a timestamp, the <pdelayRespEventIngressTimestamp> or <pdelayRespEventEgressTimestamp>, based on the value of the Timestamping Clock.

PTP event messages shall be assigned the timestamps defined above as specified in 7.3.4.

7.3.3.2 General PTP messages

The PTP general message class consists of the following PTP message types:

- a) **Announce:** Announce messages provide status and characterization information of the transmitting PTP Instance and the Grandmaster PTP Instance in the domain. This information is used by the receiving PTP Instance when executing the best master clock algorithm.
- b) **Follow_Up:** In a two-step PTP Port (see 3.1.86), the Follow_Up message communicates the value of the <syncEventEgressTimestamp> for a particular Sync message.
- c) **Delay_Resp:** The Delay_Resp message communicates the value of the <delayReqEventIngressTimestamp> to the PTP Port of the Slave PTP Instance issuing the Delay_Req message.
- d) **Pdelay_Resp_Follow_Up:** In a two-step PTP Port supporting the peer-to-peer delay mechanism, the Pdelay_Resp_Follow_Up message carries one of the following: the transmit timestamp <pdelayRespEventEgressTimestamp> generated by a PTP Port at the transmission of a Pdelay_Resp message, or the difference between the time of transmission of the Pdelay_Resp message and the time of receipt of the corresponding Pdelay_Req message.
- e) **Management:** PTP management messages communicate information and commands used to manage PTP Instances.
- f) **Signaling:** PTP Signaling messages carry information, requests, and commands between PTP Instances.

PTP general messages are not required to be timestamped.

7.3.4 Generation of PTP event message timestamps

7.3.4.1 message timestamp point

Unless otherwise specified in a transport-specific annex to this standard, the message timestamp point for a PTP event message shall be the beginning of the first symbol after the start of frame delimiter.

7.3.4.2 Timestamp generation

All PTP event messages are timestamped on egress and ingress (e.g., the <syncEventEgressTimestamp>, t_1 , etc.). The timestamp shall be the time at which the message timestamp point (see 3.1.33) passes the reference plane, marking the boundary between the PTP Instance and the network, including corrections for the latencies described in this clause, and illustrated in Figure 26.

If an implementation captures PTP event message timestamps using a point in the PTP message (the implemented message timestamp point; see 3.1.20) other than the message timestamp point, then the captured timestamps must be appropriately corrected by the time interval between the actual time of capture and the time the message timestamp point passes the reference plane. This time interval is the <messageTimestampPointLatency> illustrated in Figure 26.

In general, the timestamps provided to PTP may be captured at a point removed from the reference plane. Furthermore, the time offset from the reference plane is likely to be different for inbound and outbound PTP event messages. To meet the requirement of this subclause, the captured timestamps must be corrected for these offsets:

- The captured timestamps should be corrected partially or fully by the implementation before being provided to the PTP stack.
- The timestamps provided to the PTP stack may be corrected for the residual offsets, that is <ingressLatency> and <egressLatency>, respectively, as specified in this clause.

Failure to make these corrections might result in a time offset between the Slave Clocks and Master Clocks.

The implementation-specific corrections of the captured timestamps are specified as follows:

$$\text{<egressProvidedTimestamp>} = \text{<egressCapturedTimestamp>} + \text{<implementation-specific correction of egressLatency and messageTimestampPointLatency>}$$

$$\text{<ingressProvidedTimestamp>} = \text{<ingressCapturedTimestamp>} - \text{<implementation-specific correction of ingressLatency and messageTimestampPointLatency>}$$

Determining the values of these implementation-specific corrections is outside the scope of this standard. The desired values of the implementation-specific corrections can depend on various factors, including on the configuration of the medium and of the physical port, for example, operating mode, speed, and the type of SFP. Implementations should attempt to appropriately correct for any effect that can be taken into account during their design.

Figure 26 illustrates the message timestamp point and reference plane offsets. Based on this model, the appropriate PTP corrections are as follows:

$$\text{<egressTimestamp>} = \text{<egressProvidedTimestamp>} + \text{<egressLatency>}$$

$$\text{<ingressTimestamp>} = \text{<ingressProvidedTimestamp>} - \text{<ingressLatency>}$$

where the timestamps <egressTimestamp> and <ingressTimestamp> at the reference plane are computed from the provided timestamps by the corrections for <egressLatency> and <ingressLatency>.

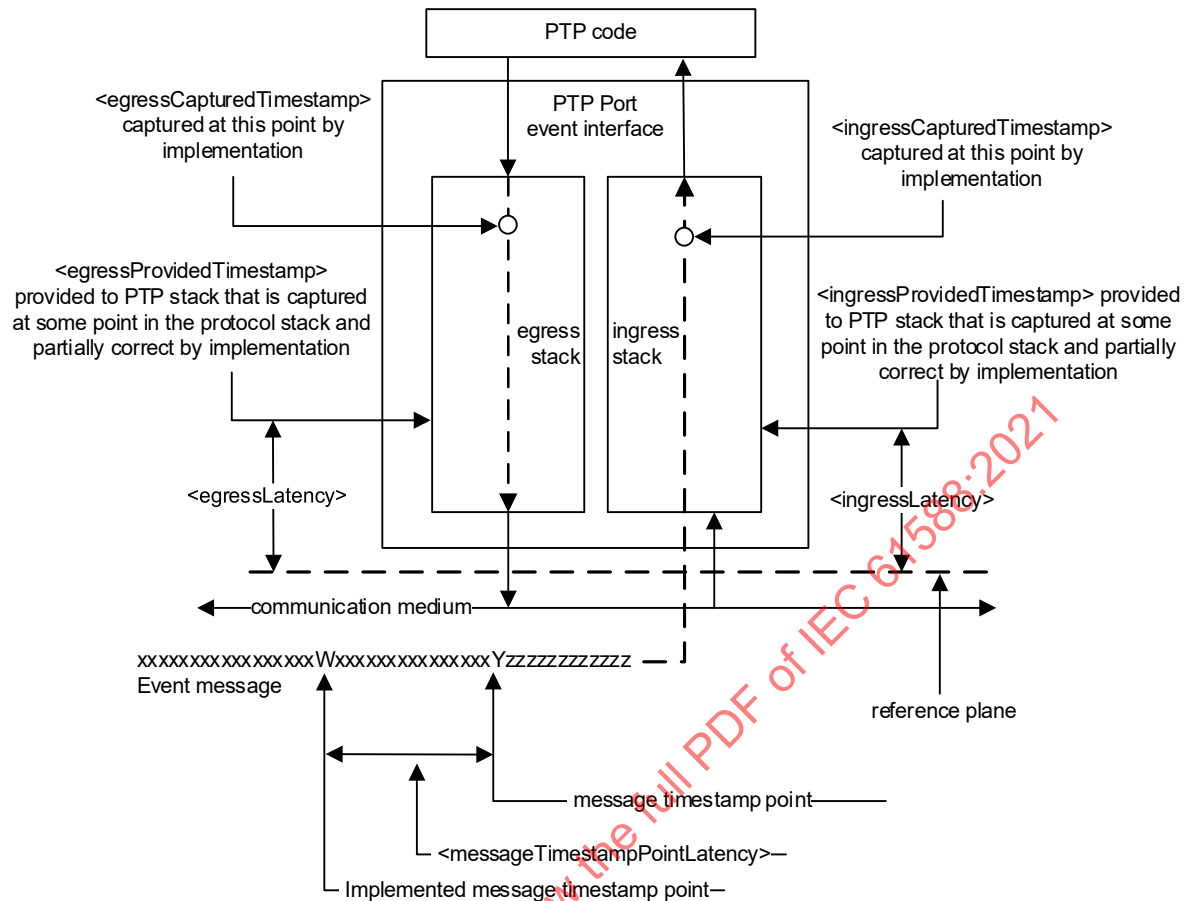


Figure 26—Definition of latency constants

NOTE 1—If the captured timestamps are not sufficiently corrected for the `<messageTimestampPointLatency>` by the implementation before being provided to the PTP, these timestamps can be still corrected in PTP by appropriately adjusting the values of `<egressLatency>` and `<ingressLatency>`.

All `<ingressTimestamp>` and `<egressTimestamp>` values shall be generated based on the Timestamping Clock (see 7.3.4.3).

NOTE 2—The correction of latencies is relative. In this subclause, three different terms are used to refer to different points at which timestamps are expressed: “captured”, “provided”, and “generated”. In essence:

- The timestamp is “captured” by the implementation at some point P1, which is possibly removed from the reference plane.
- The implementation applies an internal correction; after this correction is applied, the revised timestamp is relative to a new point P2, which is presumably closer to the reference plane than P1. The timestamp at point P2 is “provided” to the PTP protocol (stack).
- The value `<ingressLatency>`, and `<egressLatency>` are used in the PTP computations to correct timestamp at point P2. The result is relative to a new point P3, which is presumably closer to the reference plane. The timestamp at point P3 is the “generated” timestamp, that is, `<ingressTimestamp>` or `<egressTimestamp>`, which is used in the operation of the PTP protocol.

The values of <egressLatency> and <ingressLatency> may be provided by the optional data set timestampCorrectionPortDS (8.2.16). If this optional data set is implemented, the values of <egressLatency> and <ingressLatency> shall be the value of timestampCorrectionPortDS.egressLatency (8.2.16.2) and timestampCorrectionPortDS.ingressLatency (8.2.16.3), respectively. If the optional feature of 16.7 is used, then the optional data set timestampCorrectionPortDS (8.2.16) is implemented (see 16.7).

7.3.4.3 Timestamping Clock

The Timestamping Clock used in generating <ingressTimestamp> and <egressTimestamp> values of all PTP event messages of a PTP Instance should be either the Local Clock or the Local PTP Clock of the PTP Instance. PTP Instances' use of the Local Clock for timestamping some PTP event messages and the Local PTP Clock for others is permitted but not recommended unless the appropriate corrections are applied to generate timestamps.

NOTE—It is possible to design systems where the same clock is not used to timestamp all PTP event messages, for example, the Local PTP Clock for Sync and Follow_Up and a Local Clock for Pdelay_Req and Pdelay_Resp messages. If this is done, it is the responsibility of the implementation to account for errors that can be introduced due to frequency differences between the clocks.

For each type of PTP Instance, that is, Boundary Clock, Ordinary Clock, or Transparent Clock, and for each PTP event message type, that is, Sync and Delay_Req, or Sync, Pdelay_Req, and Pdelay_Resp, the designation of either the Local PTP Clock or the Local Clock as the Timestamping Clock:

- a) May be specified as a result of enabling a particular option that specifies these designations, for example, Cumulative Frequency Transfer option, otherwise
- b) May be specified in the applicable PTP Profile, otherwise
- c) Is implementation specific. However, in this case it is highly recommended:
 - 1) The same clock, i.e. the Local PTP Clock or the Local Clock, be used for timestamping all PTP event messages processed by the PTP Instance, and
 - 2) For Ordinary Clocks and Boundary Clocks, the Local PTP Clock be designated as the Timestamping Clock and
 - 3) That for Transparent Clocks, the Local PTP Clock or a syntonized Local Clock be designated as the Timestamping Clock

7.3.5 PTP message sourcePortIdentity

Each PTP message contains a sourcePortIdentity field that identifies the egress PTP Port that originated this PTP message (see 13.3.2.11).

7.3.6 PTP message types

Each PTP message contains a messageType field that indicates the type of PTP message (see 13.3.2.3).

7.3.7 PTP message sequenceId

With the exceptions noted in this subclause, each PTP Port of an Ordinary Clock or Boundary Clock shall maintain a separate sequenceId pool for each PTP message type that originates at the Ordinary Clock or Boundary Clock, and that is sent to the same destination address. With the exceptions noted here, each PTP Port on a Transparent Clock shall maintain a separate sequenceId pool for each of Signaling, PTP management, and Pdelay_Req messages that originate (i.e., that are not a retransmission of a received PTP

message) at the Transparent Clock. Multicast addresses are considered to be single destination addresses, and each unicast address is considered to be a destination address.

Except after a PTP Port state change, or the expiration or cancellation of a unicast negotiated contract, the sequenceId of the PTP message shall be one greater than the sequenceId of the previous PTP message of the same PTP message type sent to the same PTP message destination address by the transmitting PTP Port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field.

Two PTP multicast messages bearing the same values for the messageType, domainNumber, and sdoId fields and transmitted from the same transmitting protocol address of a PTP Port are considered to be two instances of the same PTP message if the values of the sequenceId fields are identical, consistent with the rollover properties defined earlier for the sequenceId. When using a unicast operation, two PTP messages bearing the same values for the messageType, domainNumber, and sdoId fields, and transmitted from the same transmitting protocol address of a PTP Port, and transmitted to the same destination protocol address, are considered to be two instances of the same PTP message if the values of the sequenceId fields are identical, consistent with the rollover properties defined earlier for the sequenceId.

Separate pools of sequenceId shall not be maintained for the following PTP message types:

- Pdelay_Resp
- Follow_Up
- Delay_Resp
- Pdelay_Resp_Follow_Up
- PTP management messages that are a response to another PTP management message

For these exceptions, the sequenceId value is specified in 13.3.2.12 and in 15.4.1.2.

7.3.8 Flag-based Indicators

7.3.8.1 unicastFlag

A TRUE value for the flag unicastFlag (see 13.3.2.8) indicates that the PTP message is transmitted as a unicast PTP message.

7.3.8.2 alternateMasterFlag

A TRUE value for the flag alternateMasterFlag (see 13.3.2.8) indicates that the PTP message is transmitted from a PTP Port not in the MASTER state.

7.3.8.3 twoStepFlag

A TRUE value for the flag twoStepFlag (see 13.3.2.8) indicates that a Sync or a Pdelay_Resp message will be followed by a Follow_Up or Pdelay_Resp_Follow_Up message, respectively. A FALSE value for the flag twoStepFlag (see 13.3.2.8) indicates that a Sync or a Pdelay_Resp message will not be followed by a Follow_Up or Pdelay_Resp_Follow_Up message, respectively.

7.4 PTP communication media

7.4.1 Network transport protocol

PTP communications occur on paths using transport protocols defined by the mapping annexes of this standard. The identification of the transport protocol in use for a communication path shall be indicated by the networkProtocol enumeration of Table 3.

Table 3—networkProtocol enumeration

Name	Value (hex)	Applicable annex
Reserved	0000	—
UDP/IPv4	0001	Annex C
UDP/IPv6	0002	Annex D
IEEE 802.3	0003	Annex E
DeviceNet	0004	Annex F
ControlNet	0005	Annex G
PROFINET	0006	Annex H
Reserved	0007-EFFF	—
Designated for assignment in a PTP Profile	F000-FFFF	—
Unknown Protocol	FFFF	—
Reserved	FFFF	—

Enumeration assignments for other transport mechanisms will be assigned when the appropriate mapping annex is added to this standard by amendment or in a future edition.

7.4.2 PTP Communication Path delay, PTP Link delay, and <delayAsymmetry>

All PTP timing messages exchanged between PTP Ports should traverse the same path of network elements (e.g., bridges or switches, including Transparent Clocks) in their transport between these ports. This recommendation helps to both minimize asymmetry and ensure that if such messages traverse a Transparent Clock with two-step PTP Port, this Transparent Clock will be able to correctly update an associated PTP message when needed.

The Precision Time Protocol requires that the transmission times of certain PTP messages be measured between a PTP Port in the MASTER state and a PTP Port in the SLAVE state and between the PTP Port in the SLAVE state and the PTP Port in the MASTER state. For the peer-to-peer delay mechanism, it is required that the transmission time between the responder and requester PTP Ports and between the requester and the responder PTP Ports be measured. Typically, these transmission times (i.e., in the two directions) are not the same. PTP characterizes the transmission times as follows:

- a) <meanDelay> which is:
 - 1) <meanPathDelay> when delay request-response mechanism is used, or
 - 2) <meanLinkDelay> when peer-to-peer delay mechanism is used
- b) <delayAsymmetry>

NOTE 1—The <meanPathDelay> applies to an individual exchange between a single PTP Port in the SLAVE state and a single PTP Port in the MASTER state. The <meanLinkDelay> applies to an individual exchange between a single requester and a single responder PTP Port.

The basis for these attributes is illustrated in Figure 27.

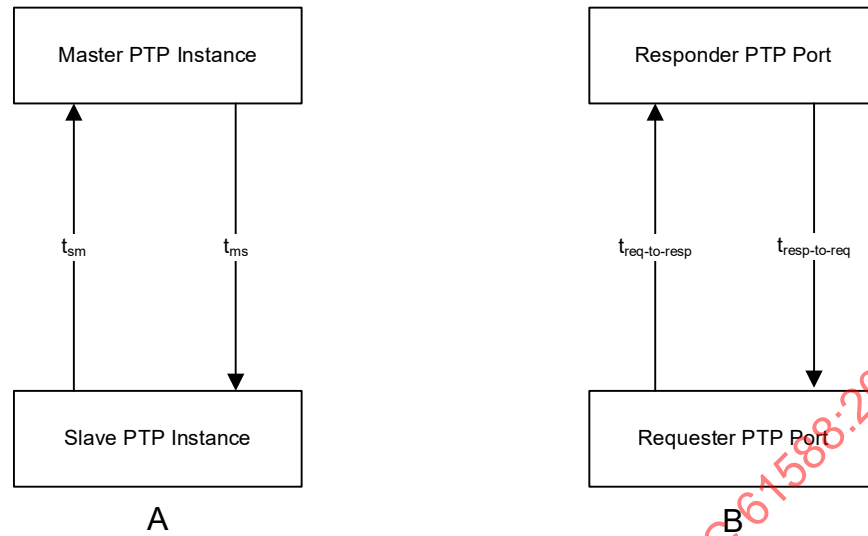


Figure 27—Delay asymmetry

The $\langle \text{meanPathDelay} \rangle$ is the mean value of t_{ms} and t_{sm} for Figure 27A, and the $\langle \text{meanLinkDelay} \rangle$ is the mean value of $t_{resp-to-req}$ and $t_{req-to-resp}$ for Figure 27B; that is, $\langle \text{meanPathDelay} \rangle = (t_{ms} + t_{sm})/2$ and $\langle \text{meanLinkDelay} \rangle = (t_{resp-to-req} + t_{req-to-resp})/2$, respectively.

NOTE 2—Although the name of $\langle \text{meanPathDelay} \rangle$ implies that the value it contains is the mean of the delay in the two directions over the PTP Communication Path, this interpretation is correct only in certain scenarios. The $\langle \text{meanPathDelay} \rangle$ is a value computed to assist in the computation of the $\langle \text{offsetFromMaster} \rangle$ to determine the correct time at the PTP Port in the SLAVE state. As such, it differs from the actual path delay by the removal of the correctionField information, which can include fractional nanoseconds from the PTP Port in the MASTER state and any residence times computed by Transparent Clocks present in the PTP Communication Path.

NOTE 3—Although the name of $\langle \text{meanLinkDelay} \rangle$ implies that the value it contains is the mean of the delay in the two directions over the PTP Link, this interpretation is correct only in certain scenarios. The $\langle \text{meanLinkDelay} \rangle$ is a value computed to assist in the computation of the $\langle \text{offsetFromMaster} \rangle$ to determine the correct time at the PTP Port in the SLAVE state. As such, it differs from the actual PTP Link delay by the removal of the correctionField information, which can include fractional nanoseconds from a responder PTP Port and any residence times computed by end-to-end Transparent Clocks present in the PTP Link (see 10.2.2.3.1).

NOTE 4—Since $\langle \text{meanPathDelay} \rangle$ and $\langle \text{meanLinkDelay} \rangle$ are values computed to assist in the computation of $\langle \text{offsetFromMaster} \rangle$, and in some cases differ from the actual path or PTP Link delays by the removal of the correction field information (see NOTE 2— and NOTE 3— in this subclause), the quantities t_{ms} and t_{sm} are, in some cases, not actual delays between a port in the MASTER state and a port in the SLAVE state, and the quantities $t_{resp-to-req}$ and $t_{req-to-resp}$ are, in some cases, not actual delays between a requester and responder port. In such cases, the delays differ from the actual delays by the removal of correctionField information.

The value of $\langle \text{delayAsymmetry} \rangle$ is required for the time transfer computations specified in this standard. The attribute $\langle \text{delayAsymmetry} \rangle$ is defined as follows by one of the following two sets of equations:

$$\begin{aligned} t_{ms} &= \langle \text{meanPathDelay} \rangle + \langle \text{delayAsymmetry} \rangle \text{ and} \\ t_{sm} &= \langle \text{meanPathDelay} \rangle - \langle \text{delayAsymmetry} \rangle \end{aligned}$$

or

$$\begin{aligned} t_{resp-to-req} &= \langle \text{meanLinkDelay} \rangle + \langle \text{delayAsymmetry} \rangle \text{ and} \\ t_{req-to-resp} &= \langle \text{meanLinkDelay} \rangle - \langle \text{delayAsymmetry} \rangle \end{aligned}$$

In other words, <delayAsymmetry> is defined to be positive when the master-to-slave or responder-to-requester propagation time is longer than the slave-to-master or requester-to-responder propagation time, respectively.

The value of <delayAsymmetry> shall be based on the model specified in this subclause. When used in the operation of the protocol, the value of <delayAsymmetry> shall be:

- c) The value calculated by the option of 16.8: “Calculation of the <delayAsymmetry> for certain media” if this option is implemented and enabled, otherwise
- d) The configured value of portDS.delayAsymmetry (8.2.15.4.8) if this data set member is implemented and configurable, otherwise
- e) The value provided in the applicable PTP Profile if specified, otherwise
- f) Implementation specific, otherwise
- g) The value 0.

If item c), item e), item f), or item g) is used, portDS.delayAsymmetry shall provide a read-only access to the value of <delayAsymmetry>. If item d) is used, portDS.delayAsymmetry shall provide configurable access to <delayAsymmetry>.

NOTE 5—When item c) is used and the value of <delayAsymmetry> is calculated dynamically using the option of 16.8, a static correction of <delayAsymmetry> can be still applied through asymmetry CorrectionPortDS.constantAsymmetry; see 16.8 and 8.2.17.2.

NOTE 6—When item d) is used, the value of portDS.delayAsymmetry is, for example, based on external measurements (out of scope of this standard) and configured via a management mechanism.

7.4.3 Medium relative delay coefficient

On a Direct PTP Link (see 3.1.8) between two PTP Instances that correct their timestamps for latency per 7.3.4, the value of the <delayAsymmetry> is nearly equal to the asymmetry introduced by the medium, that is, the medium asymmetry.

NOTE 1—The timestamps corrected per 7.3.4 are expressed at the reference plane, that is, the boundary between PTP Instance hardware and the PTP Network medium. The more precise the correction of the timestamps, the smaller the amount of hardware-related asymmetry that is included into the calculation of the <meanDelay>, where <meanDelay> = $[(t_2 - t_1) + (t_4 - t_3)]/2 = [(t_2 - t_3) + (t_4 - t_1)]/2$ and <meanDelay> is either <meanPathDelay> or <meanLinkDelay>, depending on which delay mechanism is used (see 7.4.2, 11.3 and 11.4). When timestamps are corrected perfectly and the two PTP Instances are connected using a direct medium link, the only source of the <delayAsymmetry> defined in 7.4.2 is the asymmetry introduced by the medium. Consequently, the <delayAsymmetry> is equal to the medium asymmetry. If the correction of timestamps is not perfect but sufficient, the <delayAsymmetry> is nearly equal to the medium asymmetry. In other words, it includes a negligible contribution of hardware asymmetry.

For certain bidirectional media that provide a nearly constant relation between transmission times in the two directions, such as 1000BASE-BX10 defined in IEEE Std 802.3, the attribute <delayCoefficient> (α) is defined as follows:

$$t_{ms} = (1 + \alpha) t_{sm} \quad \text{for the delay request-response mechanism, and}$$

$$t_{req-to-req} = (1 + \alpha) t_{req-to-resp} \quad \text{for the peer-to-peer delay mechanism,}$$

where t_{ms} , t_{sm} , $t_{resp-to-req}$, and $t_{req-to-resp}$ are defined in 7.4.2.

The <delayCoefficient> (α) is a relative difference between the transmission times in the two directions.

NOTE 2—The $\langle \text{delayCoefficient} \rangle$ (α) is called “relative difference” because it is the ratio of the difference between the two transmission times to one of the two transmission times, that is, $\alpha = (t_{ms} - t_{sm})/t_{sm}$ and $\alpha = (t_{resp-to-req} - t_{req-to-resp})/t_{req-to-resp}$.

The $\langle \text{delayCoefficient} \rangle$ is positive when master-to-slave or responder-to-requester transmission time is longer than the slave-to-master or requester-to-responder transmission time, respectively.

NOTE 3—The value of the $\langle \text{delayCoefficient} \rangle$ (α) on a PTP Port A of a Direct PTP Link depends not only on the type of the medium in use, but it also depends on the direction of the medium and its asymmetry with respect to the PTP Port A that uses the $\langle \text{delayCoefficient} \rangle$ (α) when it is in the Slave state (and/or it is a requester). On the other side of this Direct PTP Link, there is an opposite direction of the medium and its asymmetry with respect to the other PTP Port B. Knowing the value of the $\langle \text{delayCoefficient} \rangle$ (α) for one direction, the value of the $\langle \text{delayCoefficient} \rangle$ (α') for the opposite direction of the medium asymmetry can be calculated as follows: $\alpha' = -\alpha/(1 + \alpha)$. This value (α') can be used either by the PTP Port B on the same Direct PTP Link on which the $\langle \text{delayCoefficient} \rangle$ (α) is used for PTP Port A or by the PTP Port A when the Direct PTP Link is modified such that the direction of the medium and its asymmetry is inverted.

For the applicable media, the $\langle \text{delayCoefficient} \rangle$ (α) can be used to calculate the value of the $\langle \text{delayAsymmetry} \rangle$ as specified by the option in 16.8. The measurement of the $\langle \text{delayCoefficient} \rangle$ for certain media is described in Annex N.

If the optional feature of 16.8 is supported, then the optional data set member `asymmetryCorrectionPortDS.scaledDelayCoefficient` (8.2.17.3) is supported and provides the value of the attribute $\langle \text{delayCoefficient} \rangle$.

NOTE 4—For some media, it is possible to provide autodetection of the type and/or direction of the medium that is connected to a PTP Port. Such detection allows subsequent configuration of the value of the $\langle \text{delayCoefficient} \rangle$ (α) that is appropriate for a given medium and its direction with respect to the PTP Port being configured. The detection and configuration is best performed upon connection of the medium to the PTP Port so that the $\langle \text{delayCoefficient} \rangle$ (α) is known for further operation of the PTP Protocol, in particular, calculation of $\langle \text{delayAsymmetry} \rangle$. The configured value of $\langle \text{delayCoefficient} \rangle$ can be updated during operation to account for dynamic variations in its value, for example, due to temperature effects. Such autodetection and configuration are outside the scope of this standard but are not precluded.

7.5 PTP Ports

7.5.1 General

The PTP Instances in a PTP Node interface with the PTP Network via entities called “PTP Ports,” as shown in Figure 28.

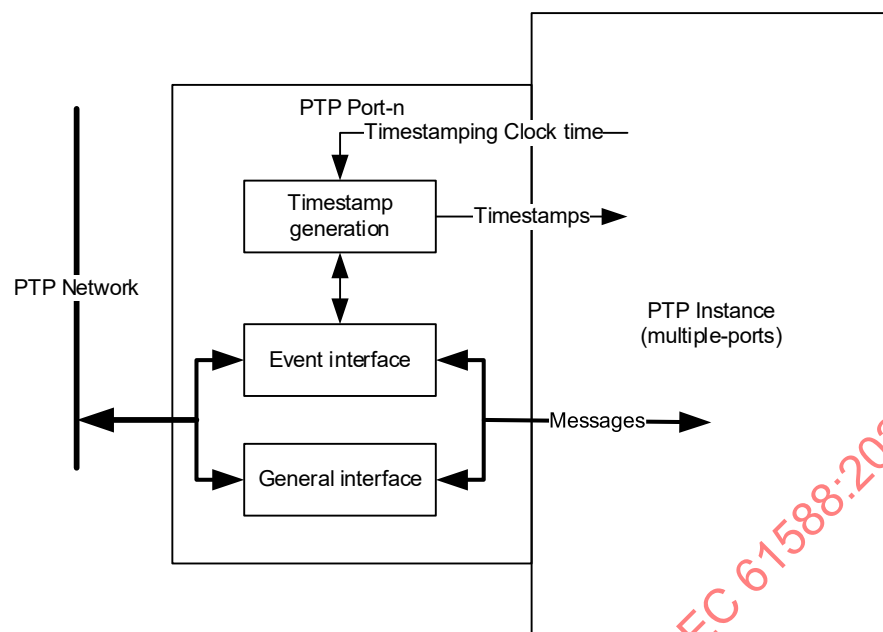


Figure 28—Model of a PTP Port

Each PTP Port on an Ordinary Clock, Boundary Clock, or Transparent Clock is modeled as supporting two interfaces, event and general, as shown in Figure 28. The model shows that timestamps are generated for PTP event messages (see 7.3.4.2) but are not required for PTP general messages.

NOTE—Figure 28 is a model. Unless otherwise stated in this standard, there is nothing precluding implementations that, for example, have a single interface, timestamp all PTP messages, and sort out the PTP event messages later.

Each PTP Port implements a single version of the protocol and uses a single transport protocol. More than one PTP Port can connect to the PTP Network via a single physical port.

The attributes of PTP Ports are described in 7.5.2 to 7.5.4.

7.5.2 portIdentity

7.5.2.1 General

A PTP Port or a Link Port (see 16.6.1) is identified by an attribute portIdentity of type PortIdentity (see 5.3.5). The values are maintained, respectively, in the portIdentity member of the portDS data set (see 8.2.15.2.1) and the cmldsLinkPortDS data set (see 16.6.4.2.2.1). A portIdentity consists of two attributes, as follows:

- portIdentity.clockIdentity
- portIdentity.portNumber

7.5.2.2 clockIdentity

7.5.2.2.1 General

The clockIdentity is an 8-octet array (see 5.3.4).

Each PTP Instance, that is, Ordinary Clock, Boundary Clock, or Transparent Clock, shall be associated with a single clockIdentity value. This value shall be unique within a PTP Network.

NOTE 1—A clockIdentity can also be associated with a PTP defined service, see for example the Common Mean Link Delay service, which issues and receives PTP messages containing a clockIdentity.

NOTE 2—If the device on which a clockIdentity is based is no longer accessible to the PTP Instance, then a new value of clockIdentity needs to be constructed and associated with the PTP Instance, and the PTP Instance needs to be re-initialized. For example, this circumstance can arise if the clockIdentity is constructed based on the EUI-48 of a NIC implemented as a line card in the PTP Node and the line card is removed. Failure to construct a new clockIdentity could result in nonuniqueness if the line card was installed in another PTP Node in the PTP Network.

The first three octets of a clockIdentity shall be either an OUI or a CID, described in the “Guidelines for Use of Extended Unique Identifier (EUI)” [B12], and can be obtained from the IEEE RA.¹²

The clockIdentity value of the PTP Instance is the same as that in the default data set, defaultDS.clockIdentity, for Ordinary Clocks, Boundary Clocks, and Transparent Clocks, as specified in 8.2.1.2.2.

NOTE 3—The CID is typically used in specifications or the implementation of devices for the purpose of identification. It identifies the organization that owns the CID-dependent sub-identifier and not necessarily the organization that defines the specification or provides the hardware. The IEEE CID listing can be obtained at: <https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries>.

NOTE 4—The Organizationally Unique Identifier (OUI) is a 24-bit identifier used as the first three octets of an IEEE 802 compliant universally (globally) unique MAC address. It may also be used by the assignee to create protocol identifiers, and other context-dependent identifiers (e.g., management information base attributes). It identifies the organization that owns the OUI-dependent subidentifier or address, which is not necessarily the organization that defines the specification or provides the hardware. Historically, the IEEE RA assignment now called “MA-L” was called an “OUI assignment.”

7.5.2.2.2 Construction of clockIdentity values

7.5.2.2.2.1 General

All clockIdentity values shall be constructed under the terms of one and only one of the following subclauses.

The clockIdentity is not a protocol address; it is an identifier.

The clockIdentity value must not be interpreted as having necessarily been formed from a protocol address.

NOTE 1—Although the clockIdentity can be constructed using a MAC address (i.e., data link layer protocol address), this is not required, and therefore, implementations of this standard cannot interpret portions of the clockIdentity as a protocol address.

NOTE 2—The terms of IEEE 1588-2008, 7.5.2.2.3 Non-IEEE EUI-64 clockIdentity values, and the use of an EUI-48

¹² IEEE Registration Authority (<http://standards.ieee.org/develop/regauth/tut/index.html/>).

to create the EUI-64 clockIdentity as described in 7.5.2.2.2 of IEEE Std 1588-2008, are not part of this edition. The terms of these two clauses under some circumstances cannot ensure the uniqueness property. The rules for constructing an EUI-64 were changed by the IEEE Registration Authority after the publication IEEE Std 1588-2008.

7.5.2.2.2.2 Construction based on an NUI-48

IEEE Std 802c defines the term NUI-48 as a 48-bit identifier that is intended to be unique within the IEEE 802 network, and it is an EUI-48, ELI-48, SAI-48, or AAI-48. The 6 octets of the NUI-48 shall be assigned in order to the six most significant octets of the 8-octet array clockIdentity, with the most significant octet of the NUI-48 assigned in order to the clockIdentity array member with index 0. The final 2 octets are inserted by the implementer such that the clockIdentity so constructed conforms to the uniqueness property of 7.5.2.2.1.

NOTE—The uniqueness requirement means that if the NUI-48 is an EUI-48, the 64 bits of the clockIdentity are not allowed to be the same as any EUI-64 that has already been assigned. This means that either

- a) The implementer needs to own the OUI [B12], MA-M, or MA-S that the EUI-48 was formed from, or
- b) The owner of the OUI, MA-M, or MA-S that the EUI-48 was formed from has given the implementer the sole right to the clockIdentities formed based on this subclause.

7.5.2.2.2.3 Construction based on an NUI-64

IEEE Std 802c defines the term NUI-64 as a 64-bit identifier that is intended to be unique within the IEEE 802 network, and is an EUI-64, SAI-64, or AAI-64. The 8 octets of the NUI-64 shall be assigned in order to the 8-octet array clockIdentity, with the most significant octet of the NUI-64 assigned to the clockIdentity array member with index 0.

7.5.2.2.3 Reserved clockIdentity value

The clockIdentity value of all ones shall designate all PTP Instances in a domain.

7.5.2.3 portNumber

The port numbers for the PTP Ports on a PTP Instance shall be mutually distinct integers between 1 and FFFE₁₆, inclusive.

The all-ones portNumber, with value FFFF₁₆, is used as the “all-ports” indicator in PTP management messages (see 15.3.1) and in Signaling PTP messages (see 13.12.1). The all-zeros portNumber value is used in the data set comparison between portIdentity and clockIdentity (see 7.5.2.4 and Table 29). The all-zeros portNumber value may also be used to represent a NULL portNumber value; for example, it can represent an uninitialized or invalid portNumber value.

7.5.2.4 Ordering of clockIdentity and portIdentity values

Two clockIdentity values X and Y are compared as follows:

- a) If every octet in X is equal to the corresponding octet in Y, then $X = Y$.
- b) Otherwise, consider the most significant position in which the octets differ, and treat the octets in that position as unsigned integers. If the octet belonging to X is smaller than the octet belonging to Y, then $X < Y$; otherwise $X > Y$.

Two portIdentities A and B of type PortIdentity with attributes clockIdentity and portNumber are compared

as follows:

- c) If A.clockIdentity is less than B.clockIdentity, then $A < B$.
- d) Otherwise, if A.clockIdentity is greater than B.clockIdentity, then $A > B$.
- e) Otherwise, if the value of A.portNumber is less than the value of B.portNumber, then $A < B$.
- f) Otherwise, if the value of A.portNumber is greater than the value of B.portNumber, then $A > B$.
- g) Otherwise, $A = B$.

A portIdentity P of type PortIdentity with attributes clockIdentity and portNumber and a clockIdentity C of type ClockIdentity are compared as follows:

- h) If P.clockIdentity is less than C.clockIdentity, then $P < C$.
- i) Otherwise, $P > C$.

NOTE—Regarding this last case, see Table 29 for how a port number of zero is used in the data set comparison algorithm.

7.5.2.5 Determination of PTP Port one or two-step semantics

Whether a PTP Instance is a one, two, or mixed-step PTP Instance and whether a PTP Port is a one-step PTP Port or two-step PTP Port are implementation dependent. In mixed-step PTP Instances, implementations may provide means to cause a PTP Port to implement one-step or two-step PTP Port semantics. Applicable PTP Profiles and some options, for example, the cumulative frequency transfer method of 16.10, may require that a PTP Port that is capable of one-step PTP Port semantics, implement two-step PTP Port semantics for all or a subset of PTP message types.

NOTE A one-step PTP Port modifies PTP message fields, for example, the correctionField, on-the-fly as the PTP message is being transmitted. Therefore, although a one-step PTP Port could implement two-step PTP Port semantics or be reconfigured to be a two-step PTP Port, the converse is not necessarily true and depends on the underlying hardware support.

7.5.3 Path delay measurement mechanism

There are two kinds of PTP Ports: stateful and stateless. Stateful PTP Ports support the state mechanism of 9.2. Stateful PTP Ports are characterized by the current state of the PTP state machine associated with the PTP Port. Stateless PTP Ports do not support the PTP state machine and do not have this state attribute.

There are two mechanisms for measuring the propagation time (i.e., path delay) of a PTP event message. These mechanisms are as follows:

- Delay request-response mechanism (see 11.3), which measures the propagation time between two stateful PTP Ports
- Peer-to-peer delay mechanism (see 11.4), which measures the propagation time between two PTP Ports supporting the peer-to-peer delay mechanism

NOTE—The peer-to-peer delay mechanism can be supported on both stateful and stateless PTP Ports.

End-to-end PTP Ports must be used to transfer time over a PTP Communication Path where the time transfer is based on the use of PTP timing messages and the delay request-response mechanism as specified in Clause 9 through Clause 13.

Peer-to-peer PTP Ports must be used to transfer time over a PTP Link where the time transfer is based on the use of PTP timing messages and the peer-to-peer delay mechanism as specified in Clause 9 through Clause 13.

Special Ports (see 11.5) must be used to transfer time over a network segment where the time transfer is not based on the use of PTP timing messages.

7.5.4 PTP versions

Major revisions shall be indicated by the attribute `versionPTP` and minor revisions by the attribute `minorVersionPTP`. The `versionPTP` and `minorVersionPTP` jointly indicate the version of this standard implemented by the PTP Instance issuing a PTP message. For this edition of the standard, the `versionPTP` attribute has the value 2 and the `minorVersionPTP` attribute has the value 1. See 19.2 for compatibility specifications based on the PTP version.

NOTE—A change in the value of `versionPTP` indicates that the formats or semantics of PTP messages or the operation of the protocol has changed such that earlier versions will not correctly interpret the newer version PTP messages. A change of only the `minorVersionPTP` indicates PTP Instances conformant to previous editions that share the same `versionPTP` value can correctly parse fields existing in the current edition. The earlier PTP Instances might not benefit from the changes introduced in the current edition. Note that the `minorVersionPTP` attribute is carried in a formerly reserved field and will therefore be ignored by PTP Instances conformant to previous editions of this standard. Since this field had the value 0 in earlier editions, it will be correctly interpreted by PTP Instances conformant to this edition. Although out of the scope of the current edition, future editions might specify autonegotiation or other techniques to enable the inclusion of PTP Instances implemented with earlier versions of the standard.

7.6 PTP Instance characterization

7.6.1 PTP Instance type

There are four types of PTP Instances, as follows:

- Ordinary Clock
- Boundary Clock
- End-to-end Transparent Clock
- Peer-to-peer Transparent Clock

Each PTP Instance is identified by a `clockIdentity` attribute (see 7.5.2.2).

Each PTP Port on a PTP Instance is identified by a `portNumber` attribute (see 7.5.2.3).

In addition, Ordinary Clocks and Boundary Clocks are characterized by the following attributes:

- `priority1`
- `priority2`
- `clockClass`
- `clockAccuracy`
- `timeSource`
- `offsetScaledLogVariance`
- `numberPorts`

Ordinary Clocks and Boundary Clocks may keep statistics on the performance of their parent using the following attributes:

- observedParentOffsetScaledLogVariance
- observedParentClockPhaseChangeRate

7.6.2 PTP Instance attributes

7.6.2.1 General requirements

The clockQuality (clockClass, clockAccuracy, and offsetScaledLogVariance) of a PTP Instance varies depending on its operating state, for example, start-up, steady-state, holdover, and on its environment, for example, temperature. A PTP Instance shall upgrade or degrade its clockQuality depending on its current synchronization state, environmental conditions, and the following logic.

If the holdoverUpgrade option of 16.4 is disabled on a PTP Instance, that is, holdoverUpgradeDS.enable is FALSE; then if the characteristics of the PTP Instance change such that the clockQuality designation no longer applies, the PTP Instance shall:

- a) If possible, upgrade or degrade the members of its clockQuality in such a way as to correctly specify the current PTP Instance characteristics, or
- b) Otherwise, if item a) is not possible:
 - 1) If option 17.6 is inactive, that is, the value of defaultDS.externalPortConfigurationEnabled is FALSE, place all PTP Ports in the FAULTY state, or
 - 2) If option 17.6 is active, that is, the value of defaultDS.externalPortConfigurationEnabled is TRUE, take implementations specific actions

If the holdoverUpgrade option of 16.4 is enabled on a PTP Instance, that is, holdoverUpgradeDS.enable is TRUE, then, the PTP Instance shall:

- c) If possible, ensure that the clockQuality specifications of the PTP Instance adhere to the specifications of 16.4, or
- d) Otherwise, if item c) is not possible:
 - 1) If option 17.6 is inactive, that is, the value of defaultDS.externalPortConfigurationEnabled is FALSE, place all PTP Ports in the FAULTY state, or
 - 3) if option 17.6 is active, that is, the value of defaultDS.externalPortConfigurationEnabled is TRUE, take implementations specific actions

7.6.2.2 clockIdentity

The clockIdentity value for a PTP Instance or PTP Management Node shall be as specified in 7.5.2.2.

7.6.2.3 priority1

The attribute priority1 is used in the execution of the best master clock algorithm (see 9.3.2). Lower values take precedence. The default initialization value of priority1 is specified in the applicable PTP Profile. The value of priority1 shall be configurable to any value in the range 0 to 255 unless restricted by limits established by the applicable PTP Profile.

NOTE—The operation of the best master clock algorithm selects PTP Instances from a set with a lower value of priority1 over PTP Instances from a set with a greater value of priority1.

7.6.2.4 priority2

The attribute priority2 is used in the execution of the best master clock algorithm (see 9.3.2). Lower values take precedence. The default initialization value of priority2 is specified in the applicable PTP Profile. The value of priority2 shall be configurable to any value in the range 0 to 255 unless restricted by limits established by the applicable PTP Profile.

NOTE—In the event that the operation of the best master clock algorithm fails to order the PTP Instances based on the values of priority1, clockClass, clockAccuracy, and scaledOffsetLogVariance, the priority2 attribute allows the creation of up to 256 priorities to be evaluated before the tiebreaker. The tiebreaker is based on the clockIdentity.

7.6.2.5 clockClass

The clockClass attribute of an Ordinary Clock or Boundary Clock denotes the traceability, synchronization state and expected performance of the time or frequency distributed by the Grandmaster PTP Instance. The interpretation and allowed values of clockClass shall be based on the definitions in Table 4. The clockClass is one of the attributes that characterizes a PTP Instance for the purpose of the best master clock algorithm (see 9.3.2).

NOTE 1—The clockClass number ranges 68 through 122 and 133 through 170 are designated for assignment in an alternate PTP Profile. For example, it is expected that these ranges will be used by PTP Profiles defining applications that distribute only frequency.

NOTE 2—A PTP Instance that is designed to synchronize its Local PTP Clock to the current Master Clock whenever it is not selected as a Master PTP Instance never sets its clockClass number to be less than 128. For example, a PTP Instance, with the time set by a user using the PTP management message TIME (see 15.5.3.2.1) and that is subsequently synchronized to the current Master Clock, preferably sets the clockClass number to 187 instead of 6 or 7. Also see Table 6 for a description of “HAND_SET”.

NOTE 3—The clockClass number range 216 through 232 is expected to be used by PTP Profiles that require PTP Instances to be given preference based on some application-specific precedence. For example, controllers might take precedence over sensors in an industrial application.

Unless a PTP Instance is specifically designed to maintain clock accuracy requirements during circumstances that result in the execution of the POWERUP event (see 9.2.6.2), the execution of this event shall preclude assigning a clockClass value of 6, 7, 13, or 14.

In a given domain, PTP Instances with clockClass values less than 128 should be inherently at least as stable (low variance) as any PTP Instance with a clockClass value greater than its own.

Table 4—clockClass specifications

clockClass (decimal)	Specification
0	Reserved for assignment by the IEEE 1588 Working Group to enable compatibility with future versions.
1–5	Reserved.
6	Shall designate a PTP Instance that is directly, that is, using a Source Dependent block (see 6.5.2), synchronized to a primary reference time source. The timescale distributed shall be PTP. Using the default BMCA of 9.3.2, a PTP Instance with clockClass 6 will not be a Slave PTP Instance to another PTP Instance in the domain.
7	Shall either: a) Designate a PTP Instance that has previously been designated as clockClass 6 but that has lost the ability to synchronize to a primary reference time source and is in holdover mode and within PTP profile designated holdover specifications, or b) Designate a PTP Instance with clockClass 7 based on the holdoverUpgrade option of 16.4 The timescale distributed shall be PTP. Using the default BMCA of 9.3.2, a clockClass 7 PTP Instance will not be a Slave PTP Instance to another PTP Instance in the domain.
8	Reserved.
9–10	Reserved for assignment by the IEEE 1588 Working Group, to enable compatibility with future versions.
11–12	Reserved.
13	Shall designate a PTP Instance that is directly, that is, using a Source Dependent block (see 6.5.2), synchronized to an application-specific source of time. The timescale distributed shall be ARB. Using the default BMCA of 9.3.2, a PTP Instance with clockClass 13 will not be a Slave PTP Instance to another PTP Instance in the domain.
14	Shall either: a) Designate a PTP Instance that has previously been designated as clockClass 13 but that has lost the ability to synchronize to an application-specific source of time and is in holdover mode and within PTP profile designated holdover specifications, or b) Designate a PTP Instance with clockClass 14 based on the holdoverUpgrade option of 16.4 The timescale distributed shall be ARB. Using the default BMCA of 9.3.2, a PTP Instance with clockClass 14 will not be a Slave PTP Instance to another PTP Instance in the domain.
15–51	Reserved.
52	Degradation alternative A: Shall either designate a PTP Instance formerly having clockClass 7: a) That is not within the designated holdover specification of the applicable PTP Profile, or b) Based on the specifications of the Holdover upgrade option 16.4 Using the default BMCA of 9.3.2, a clockClass 52 PTP Instance will not be a Slave PTP Instance to another PTP Instance in the domain.
53–57	Reserved.
58	Degradation alternative A: Shall either designate a PTP Instance formerly having clockClass 14: a) That is not within the designated holdover specification of the applicable PTP Profile, or b) Based on the specifications of the Holdover upgrade option 16.4. Using the default BMCA of 9.3.2, a PTP Instance with clockClass 58 PTP Instance will not be a Slave PTP Instance to another PTP Instance in the domain.
59–67	Reserved.
68–122	For use by alternate PTP Profiles.
123–127	Reserved.
128–132	Reserved.

clockClass (decimal)	Specification
133–170	For use by alternate PTP Profiles.
171–186	Reserved.
187	Degradation alternative B: Shall either designate a PTP Instance formerly having clockClass 7: a) That is not within the designated holdover specification of the applicable PTP Profile, or b) Based on the specifications of the Holdover upgrade option 16.4. Using the default BMCA of 9.3.2, a clockClass 187 PTP Instance can be a Slave PTP Instance to another PTP Instance in the domain.
188–192	Reserved.
193	Degradation alternative B: Shall either designate a PTP Instance formerly having clockClass 14: a) That is not within the designated holdover specification of the applicable PTP Profile, or b) Based on the specifications of the Holdover upgrade option 16.4. Using the default BMCA of 9.3.2, a clockClass 193 PTP Instance can be a Slave PTP Instance to another PTP Instance in the domain.
194–215	Reserved.
216–232	For use by alternate PTP Profiles.
233–247	Reserved.
248	Default. This clockClass shall be used if none of the other clockClass definitions apply.
249–250	Reserved.
251	Reserved.
252–254	Reserved.
255	Shall be the clockClass of a slave-only PTP Instance (see 9.2.2.1).

NOTE 4—Table 4 lists two possible variants (degradation alternative A and degradation alternative B) for the clockClass of a PTP Instance whose clock is in holdover and no longer meets the designated holdover specification (e.g., 52 or 187 for a clock degrading from clockClass 7).

Unless otherwise specified in the applicable PTP Profile, degradation alternative B shall be selected.

7.6.2.6 clockAccuracy

The clockAccuracy is one of the attribute that characterizes a PTP Instance for the purpose of the best master clock algorithm (see 9.3.2). The value of clockAccuracy shall be taken from the enumeration in Table 5. The value of this attribute shall be conservatively estimated by the PTP Instance to a precision consistent with the value of the selected clockAccuracy and of the next lower enumerated value, for example, for clockAccuracy = 23_{16} , between 250 ns and 1000 ns. This estimate shall be based on the timeSource attribute (7.6.2.8), the elapsed time since last synchronized to this time source, and the holdover specifications of the PTP Instance. If the information to determine this estimate is not available, then the enumerated value Unknown shall be used.

The clockAccuracy indicates the expected accuracy of a PTP Instance when it is the Grandmaster PTP Instance, or in the event it becomes the Grandmaster PTP Instance.

The ordering of clockAccuracy in the operation of the best master clock algorithm (see 9.3.2) is specified as follows. When comparing clockAccuracy values, PTP Instance A shall be deemed better than PTP Instance B if the value of the clockAccuracy of A is lower than that of B.

NOTE—The range from 80_{16} to FD_{16} is designated for assignment by alternate PTP Profiles. It is expected that this range will be used by PTP Profiles defining applications that distribute only frequency to define accuracy specifications appropriate for frequency distribution.

Table 5—clockAccuracy enumeration

Value (hex)	Specification
00 to 16	Reserved
17	The time is accurate to within 1 ps
18	The time is accurate to within 2.5 ps
19	The time is accurate to within 10 ps
1A	The time is accurate to within 25 ps
1B	The time is accurate to within 100 ps
1C	The time is accurate to within 250 ps
1D	The time is accurate to within 1 ns
1E	The time is accurate to within 2.5 ns
1F	The time is accurate to within 10 ns
20	The time is accurate to within 25 ns
21	The time is accurate to within 100 ns
22	The time is accurate to within 250 ns
23	The time is accurate to within 1 μ s
24	The time is accurate to within 2.5 μ s
25	The time is accurate to within 10 μ s
26	The time is accurate to within 25 μ s
27	The time is accurate to within 100 μ s
28	The time is accurate to within 250 μ s
29	The time is accurate to within 1 ms
2A	The time is accurate to within 2.5 ms
2B	The time is accurate to within 10 ms
2C	The time is accurate to within 25 ms
2D	The time is accurate to within 100 ms
2E	The time is accurate to within 250 ms
2F	The time is accurate to within 1 s
30	The time is accurate to within 10 s
31	The time is accurate to >10 s
32 to 7F	Reserved
80 to FD	Designated for assignment by alternate PTP Profiles
FE	Unknown
FF	Reserved

7.6.2.7 offsetScaledLogVariance

The offsetScaledLogVariance is one of the attributes that characterizes a PTP Instance for the purpose of the best master clock algorithm (see 9.3.2). The computation of the value of offsetScaledLogVariance is specified in 7.6.3.5.

7.6.2.8 timeSource

This information-only attribute indicates the immediate source of time used by the Grandmaster PTP Instance, that is, the nature of the Clock Source block of Figure 5 and Figure 6 of the layered model. For example, 20₁₆ indicates that the immediate source of time for the domain is a GNSS receiver. However, the GNSS system itself is a time distribution system that distributes time maintained by a national laboratory. The value is not used in the selection of the Grandmaster PTP Instance. The values shall be as specified in Table 6. These values represent categories. For example, the GNSS entry would include not only the GPS system of the U.S. Department of Defense but also of the European Galileo, Russian GLONASS, and the Chinese Beidou systems, as well as other present and future satellite-based timing systems.

Table 6—timeSource enumeration

Value (hex) ^a	timeSource	Description
10	ATOMIC_CLOCK	Any PTP Instance that is based on an atomic resonance for frequency, or a PTP Instance directly connected to a device that is based on an atomic resonance for frequency.
20	GNSS	Any PTP Instance synchronized to a satellite system that distributes time and frequency.
30	TERRESTRIAL_RADIO	Any PTP Instance synchronized via any of the radio distribution systems that distribute time and frequency.
39	SERIAL_TIME_CODE	Any PTP Instance synchronized via any of the serial time code distribution systems that distribute time and frequency, for example, IRIG-B.
40	PTP	Any PTP Instance synchronized to a PTP-based source of time external to the domain.
50	NTP	Any PTP Instance synchronized via NTP or Simple Network Time Protocol (SNTP) servers that distribute time and frequency.
60	HAND_SET	Used for any PTP Instance whose time has been set by means of a human interface based on observation of a source of time to within the claimed clock accuracy.
90	OTHER	Other source of time and/or frequency not covered by other values.
A0	INTERNAL_OSCILLATOR	Any PTP Instance whose frequency is not based on atomic resonance, and whose time is based on a free-running oscillator with epoch determined in an arbitrary or unknown manner.
F0 to FE	Designated for assignment by alternate PTP Profiles	
FF	Reserved	

NOTE 1—The value of clockAccuracy needs to be consistent with the uncertainty value used for traceability (see 3.1.80). A PTP Profile can specify requirements for the uncertainty value. The values for clockClass, clockAccuracy, and timeSource need to be consistent. For example, a class 6 atomic clock synchronized directly to the GPS system might claim an accuracy of 25 ns, while the same atomic clock not synchronized to GPS but set via a user interface by a user observing a National Institute of Standards and Technology (NIST) server via the Web might claim to be class 6 but with an accuracy of 10 s.

NOTE 2—The range from F0₁₆ to FE₁₆ is designated for assignment by alternate PTP Profiles. It is expected that this range will be used by PTP Profiles defining applications that distribute only frequency to define the nature of sources appropriate for frequency distribution.

NOTE 3—These designations might not carry over a power-fail restart, but in any case, they need to reflect the current status of the PTP Instance. For example, a simple quartz oscillator at turn on would be INTERNAL_OSCILLATOR. If later the epoch was set by hand, it would be HAND_SET, whereas if it later synchronized to GPS, it would be GPS. If it had a battery-backed up real-time clock, this status could survive a power-fail restart although the clockAccuracy and perhaps clockClass would be degraded.

^a All unused values in Table 6 are reserved.

7.6.2.9 numberPorts

The attribute, numberPorts, shall indicate the number of PTP Ports on the PTP Instance.

7.6.3 PTP variance

7.6.3.1 General

Two variance estimates, as specified in 7.6.3.2, characterize Ordinary Clock and Boundary Clocks in a PTP Network. They are as follows:

- Each such PTP Instance shall maintain an estimate, the `offsetScaledLogVariance` (see 7.6.3.5), of its inherent precision. This is the precision of the timestamps included in PTP messages issued by the PTP Instance when it is not synchronized to the Local PTP Clock of another PTP Instance using the protocol. The `offsetScaledLogVariance` depends on both the stability of the Local Clock of the PTP instance and any error introduced in the timestamping process.
- If such a PTP Instance, `clock_A`, is synchronized to another using the PTP protocol, it may maintain an estimate, the `observedParentOffsetScaledLogVariance` (see 7.6.4.3), of the precision of the clock to which it is synchronized as observed by `clock_A`.

7.6.3.2 Variance algorithm

The PTP variance, from which `offsetScaledLogVariance` and `observedParentOffsetScaledLogVariance` are derived, is based on the theory of Allan deviation.

The Allan deviation $\sigma_y(\tau)$ is estimated as follows (see ITU-T Recommendation G.810 [B31] and Bregni [B3]) in Equation (3).

$$\sigma_y(\tau) = \left[\frac{1}{2(N-2n)\tau^2} \sum_{k=1}^{N-2n} (x_{k+2n} - 2x_{k+n} + x_k)^2 \right]^{1/2} \quad (3)$$

where τ is the observation interval (see ITU-T Recommendation G.810 [B31] and Bregni [B3]) and is equal to $n\tau_0$, τ_0 is the sampling interval, n is the number of sampling intervals in one observation interval, N is the total number of data samples, and x_k , x_{k+n} , and x_{k+2n} are time residual measurements made at times t_k , $t_{k+n} = t_k + \tau$, and $t_{k+2n} = t_k + 2\tau$. The term “residual” implies that any consistent systematic effects have been removed from the data.

NOTE 1—See ITU-T Recommendation G.810 [B31] and Bregni [B3] for additional details on the Allan deviation.

The Allan deviation is a second-order statistic on the variation of the frequency of the oscillator used as the basis of the time base.

The PTP variance, $\sigma_{PTP}^2(\tau)$, is defined by Equation (4).

$$\sigma_{PTP}^2(\tau) = \frac{\tau^2}{3} \sigma_y^2(\tau) \quad (4)$$

An unbiased estimate of the PTP variance shall be computed as follows in Equation (5).

$$\sigma_{PTP}^2(\tau) = \frac{1}{6(N-2n)} \sum_{k=1}^{N-2n} (x_{k+2n} - 2x_{k+n} + x_k)^2 \quad (5)$$

where x_k , x_{k+n} , and x_{k+2n} are time residual measurements, made at times t_k , $t_{k+n} = t_k + \tau$, and $t_{k+2n} = t_k + 2\tau$, respectively, between the time provided by the measured clock and a local reference clock, N is the number of data samples, and n is the number of sampling intervals in one observation interval. For a PTP variance, the quantity τ , the observation interval, shall be the value defined in the applicable PTP Profile. Subclauses 8.2.1.3.1.4 and 8.2.3.4 specify variances to be computed using the specifications in this subclause. If these variances are computed during execution, data are only available in multiples of the syncInterval (see 7.7.2.3). In this case, τ should be a multiple of the syncInterval (if τ is not a multiple of the syncInterval, some form of interpolation will be necessary).

Implementations may compute a conservative estimate of the PTP variance rather than compute the exact value given here. Note that this might be necessary in implementations with limited computational or memory resources.

The offsetScaledLogVariance is one of the attributes used by the BMCA to select the Grandmaster PTP Instance. In order for values of defaultDS.clockQuality.offsetScaledLogVariance of different PTP Instances to be comparable, the PTP Profile should specify the value of observation interval τ for use in all PTP Instances of that profile. The observation interval is preferred to be chosen to reflect a time interval over which stability of the transported clock signal is important, for the respective application(s).

NOTE 2—The dependence of the Allan deviation on the sample period provides information on the type of the underlying noise processes. The Allan deviation is not sensitive to constant offsets in time or in frequency, even though those offsets can be important in some applications of this standard. In addition, the Allan deviation does not provide a useful diagnostic when the noise spectrum contains “bright lines”—power-line-induced variations at 60 Hz, for example. Finally, the Allan deviation is defined as an average over the ensemble of observations, and it is most useful when the data are statistically stationary (see ITU-T Recommendation G.810 [B31] and Bregni [B3]). The deviation does not provide a good measure of the frequency or amplitude of occasional glitches, even though such events can be important in some applications of this standard.

NOTE 3—In the 2008 edition of this standard, the expressions for $\sigma_y(\tau)$ and $\sigma_{PTP}^2(\tau)$ are for the special case where the observation interval equals the sampling interval, that is, $\tau = \tau_0$ and $n = 1$.

7.6.3.3 Variance representation

PTP variances shall be represented as follows:

- An estimate of the variance, σ_{PTP}^2 specified in 7.6.3.2 is computed in units of seconds squared.
- The logarithm to the base 2 of this estimate is computed. The computation of the logarithm need not be more precise than the precision of the estimate of the variance.
- The logarithm is multiplied by 2^8 to produce a scaled value.
- This scaled value is modified per the hysteresis specification of this subclause to produce the reported value.
- The reported value is represented as a 2’s complement Integer16. The value 8000_{16} is added to the reported value represented in this form, and any overflow is ignored. The result, that is, the offset scaled reported value, is cast as a UInteger16.
- This offset scaled reported value, represented as UInteger16, shall be the value of the log variances specified in 7.6.3.1.

NOTE 1—For example, suppose the PTP variance value is $1.414 \times 2^{-73} = 1.497 \times 10^{-22} \text{ s}^2$. Therefore, $\log_2(1.414 \times 2^{-73}) = -73 + 0.5 = -72.5$. If this were expressed as an Integer16, it would truncate to -72 . To retain some precision, the value -72.5 is scaled by 2^8 to yield a scaledLogVariance of -18560 , that is, $B780_{16}$, which retains 8 bits more precision. To this is added 8000_{16} to yield the offset scaled reported value 3780_{16} .

NOTE 2—The smallest variance that can be represented is 2^{-128} or $\sim 3 \times 10^{-39} \text{ s}^2$, which results in an offsetScaledLogVariance of 0000_{16} . The maximum variance that can be represented is $\sim 2^{+127.99219}$, which results in an offsetScaledLogVariance of $FFFE_{16}$.

NOTE 3—This representation ensures that the ordering of variances algorithm of 7.6.3.4 produces identical results in all implementations. This result cannot be guaranteed with a floating-point representation.

The largest possible positive number, $FFFF_{16}$, for the offsetScaledLogVariance attribute shall indicate that the variance is either too large to be represented or has not been computed.

Since variance values are used in the selection of the best master clock (see 9.3.2), implementations in which variance values are computed during operation shall include hysteresis in the estimation of the variances to preclude thrashing in the process of selecting the Master PTP Instance. The magnitude of this hysteresis, applied to the logarithm to base 2 of the reported estimates scaled by 2^8 , shall be $PTP\_SCALED\_LOG\_VARIANCE\_HYSTERESIS$, as shown in Figure 29. This hysteresis shall be applied to the scaled values of the logarithm of the estimate, produced by item c) in this subclause and illustrated in Figure 29. The result is the offset scaled values of the logarithm of the estimate (i.e., the ordinate in Figure 29), which is reported and used in the computations of the best master clock algorithm (see 9.3.2.2). Sufficient local state needs to be maintained to allow correct implementations of the hysteresis properties for both increasing and decreasing trends in the actual variance estimate. The value of $PTP\_SCALED\_LOG\_VARIANCE\_HYSTERESIS$ is 2^7 .

NOTE 4—A value of 2^7 corresponds to a change in the value of the actual $\log_2(\text{actual estimate})$ of $1/2$. Fluctuations in computed $\log_2(\text{estimated variance})$ below $1/2$ are not to be reported because of this hysteresis requirement.

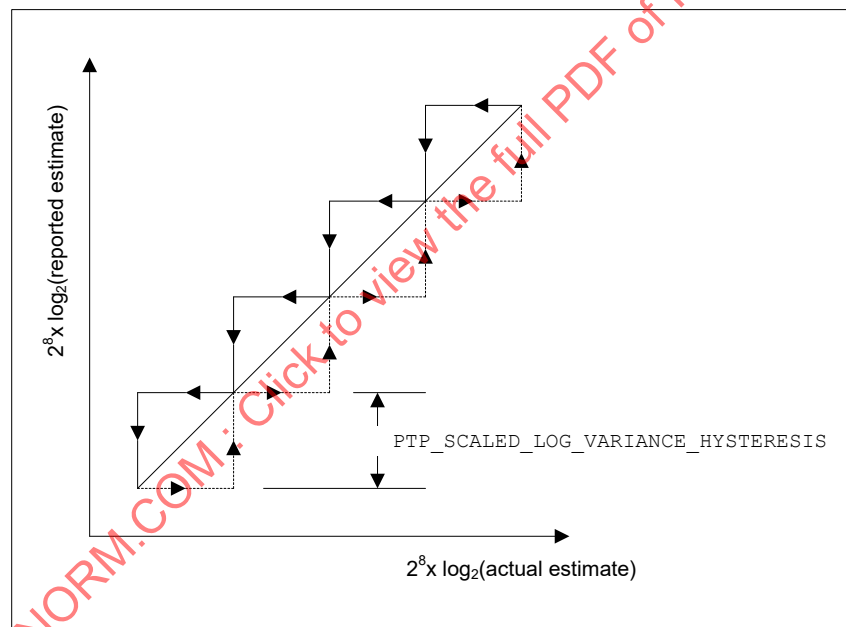


Figure 29—Scaled log variance hysteresis

7.6.3.4 Ordering of variances

The ordering of variances shall be computed on the offset, scaled, logarithmic representation of the variance.

7.6.3.5 Computation of defaultDS.offsetScaledLogVariance

The value of defaultDS.offsetScaledLogVariance shall be an estimate of the variations of the Local PTP Clock when it is not synchronized to another Local PTP Clock using the protocol, as measured by comparison with a suitable reference clock. The reference clock can be an atomic clock, a GPS receiver, a stable local oscillator, a suite of clocks synchronized via NTP, and so on. These sources can contribute to the variance estimate.

The value of defaultDS.offsetScaledLogVariance shall be:

- A constant expressing a worst case for all operating states and conditions, or
- A variable derived from measured or modeled behavior of the components of the Local PTP Clock and its environment, or
- A constant taken from a set of worst-case values corresponding to the operating states

The value of defaultDS.offsetScaledLogVariance shall be estimated consistent with, and represented as described in, 7.6.3.2 and 7.6.3.3.

The value of defaultDS.offsetScaledLogVariance is the variance applicable to an ensemble of measurements including error contributions from:

- Synchronization to its reference clock
- Variation in clock phase change rate and noise characteristics of the local oscillator
- Sampling quantization errors, fluctuations in inbound and outbound latency, and other fluctuations of the Local PTP Clock

7.6.4 Parent PTP Instance statistics

7.6.4.1 General

A PTP Instance in the SLAVE state may maintain statistics on the observed performance of its Parent PTP Instance. The two optional statistics are as follows:

- parentDS.observedParentOffsetScaledLogVariance
- parentDS.observedParentClockPhaseChangeRate

7.6.4.2 parentDS.parentStats

The value of parentDS.parentStats shall indicate whether the values of parentDS.observedParentOffsetScaledLogVariance and parentDS.observedParentClockPhaseChangeRate have been measured and are valid. A TRUE value shall indicate valid data. A FALSE value shall indicate that either the measurements have not been made or that the validity of the data is unknown.

7.6.4.3 parentDS.observedParentOffsetScaledLogVariance

The parentDS.observedParentOffsetScaledLogVariance of a Local PTP Clock shall be the variance of the phase offset of the Local PTP Clock of the Parent PTP Instance as measured with respect to the Local PTP Clock of this PTP Instance. This measurement is optional.

The value of `parentDS.observedParentOffsetScaledLogVariance` shall be computed and represented per 7.6.3.2 and 7.6.3.3.

7.6.4.4 `parentDS.observedParentClockPhaseChangeRate`

The `parentDS.observedParentClockPhaseChangeRate` shall be an estimate of the phase change rate of the Local PTP Clock of the Parent PTP Instance, as measured by the Local PTP Clock of this PTP Instance (see 3.1.44). The reported value shall be the fractional frequency offset multiplied by 2^{+40} . If the estimate exceeds the capacity of its data type, this value shall be set to the largest or smallest allowable value, as appropriate. A positive sign indicates that the Local PTP Clock of the Parent PTP Instance is faster than the Local PTP Clock of the Slave PTP Instance. This measurement is optional.

NOTE 1—The minimum phase change rate that can be expressed is 2^{-40} or approximately 1 ps per s.

NOTE 2—This value is dependent on the measurement time interval used. When this value is critical for an application domain, the time interval could be specified in the applicable PTP Profile.

7.6.5 `defaultDS.numberPorts`

The value of `defaultDS.numberPorts` shall be the number of PTP Ports on the PTP Instance.

7.6.6 Obtaining timing from external sources and updating the data sets of a PTP Instance

7.6.6.1 General specification

The time, phase, frequency, and metadata describing these are referred to in 7.6.6 as “timing.” The Source Dependent block consists of one or more Source Adapter blocks and a single External Source block. Each PTP Instance may also contain a Local PTP Clock, which in some circumstances serves as the source of time for the domain (see 7.6.6.3).

When time is transferred into a domain from Clock Sources (sources of time external to the domain), only mechanisms consistent with the Source Dependent block model of Figure 5 and Figure 6 and 6.5.2.1 shall be used.

This mechanism shall operate such that:

- a) The transfer of timing into the domain shall not affect the operation of the Clock Source except as necessary to obtain the time and/or frequency and any available metadata describing the timescale and the Clock Source.
- b) The transfer of timing into the domain shall not affect the operation of the PTP protocol within the domain except:
 - 1) Based on the population of the `defaultDS.clockQuality`, `priority1` and `priority2` members, and any resultant action of the BMCA or of the mechanism of 17.6, and
 - 2) In the event that the PTP Instance is selected as the Grandmaster PTP Instance of the domain:
 - i) the time distributed by the Grandmaster PTP Instance will be either a) time provided by the selected external Clock Source or b) time based on the Local PTP Clock (see 7.6.6.2), and
 - ii) Other properties that may be changed based on the timing metadata are the: `timeTraceable` flag, `frequencyTraceable` flag, `timeProperties` data set, and

- iii) The frequency of the Timestamping Clock of the Grandmaster PTP Instance may be syntonized to an external source of frequency, for example, a telecom PRC, or the Clock Source
- 3) If specified by the applicable PTP Profile, the frequency of the Local Clock may be syntonized to an external source of frequency, for example, a telecom PRC

NOTE—Examples of Clock Sources are a GPS receiver, a calibrated atomic clock, an NTP server, and PTP operating in a different domain (see Table 6).

7.6.6.2 Source Adapter blocks

A PTP Instance may implement multiple Source Adapter blocks. Each Source Adapter block accepts timing from a single Clock Source external to the domain of the PTP Instance. The external interface of a Source Adapter block is specific to a specific external timing Clock Source.

A Source Adapter block shall provide to the interface of the External Source block, candidate clockQuality values (clockClass, clockAccuracy, and offsetScaledLogVariance) of the default data set, 8.2.1.3.1, and candidates for all the values of the time properties data set, 8.2.4. These candidate values shall characterize the specific Clock Source associated with the Source Adapter block.

The Source Adapter block shall also provide the timing signals required to synchronize the Local PTP Clock if the external Clock Source is selected as the source of time for the PTP Instance.

7.6.6.3 External Source block

The External Source block may receive timing information and signals from one or more Source Adapter blocks.

If multiple Source Adapter blocks are present, an algorithm, the source selection algorithm, within the External Source block shall select a source of timing for evaluation by the BMCA or by the mechanism of 17.6.

NOTE 1—In Figure 5 and Figure 6, each arrow pointing to the External Source Block is from a different Source Adapter block.

The source selection algorithm may select

- One or some combination of the external Clock Sources provided by one or more Source Adapter blocks, or
- The Local PTP Clock

The source selection algorithm shall be based on the known characteristics of the Local PTP Clock and the attribute values provided by the Source Adapter blocks (see 7.6.6.2).

The source selection algorithm may be specified in a Profile. In the absence of a Profile-defined algorithm, the algorithm is implementation specific.

Based on the results of the source selection algorithm, the External Source block shall populate the clockQuality values (clockClass, clockAccuracy, and offsetScaledLogVariance) of the default data set, 8.2.1.3.1, 7.6.2.5, 7.6.2.6, and 7.6.3.5, and shall be the source of the values for the update of the time properties data set if required by the terms of Table 30 and Table 33 of 9.3.5.

NOTE 2—The BMCA (see 9.3) makes use of the values of the defaultDS data set in determining whether the source of time for the PTP Instance is based on a) the external Clock Sources or the Local PTP Clock, that is, the PTP Instance is the Grandmaster PTP Instance, or b) is based on timing received from another PTP Instance in the domain via a PTP Port, that is, the PTP Instance is not the Grandmaster PTP Instance. When the value of defaultDS.externalPortConfigurationEnabled is TRUE, the BMCA is disabled and the mechanism of 17.6 might or might not use this information

7.6.7 Providing timing to external users

7.6.7.1 General specification

The time, phase, frequency, and metadata describing these are referred to in 7.6.7 as “timing.” When time is transferred out of a domain, mechanisms consistent with the Sink Dependent block model of Figure 5 and Figure 6 and 6.5.2.1.2 shall be used.

This mechanism shall operate such that:

- a) The transfer of time out of the domain shall not affect the operation of the PTP Instance providing the timing except as necessary to obtain the time and any available metadata describing the timescale, and
- b) The operation of the recipient of the time, the Clock Sink of Figure 5 and Figure 6, shall not be affected except based on the time and metadata provided

NOTE—Examples of Clock Sinks are sensors, controllers, network devices, or a PTP Instance operating in a different domain.

Each PTP Instance delivering time to a Clock Sink contains a Local PTP Clock that serves as the source of time for transfer to Clock Sinks.

The Sink Dependent block consists of one or more Sink Adapter blocks and a single External Sink block.

7.6.7.2 Sink Adapter blocks

A Sink Adapter block shall provide timing to a Clock Sink external to the PTP domain of the PTP Instance.

The Sink Adapter block interface to the External Sink block shall base the timing provided to the external Clock Sink on the clockQuality values (clockClass, clockAccuracy, and offsetScaledLogVariance) of the parentDS data set (see 8.2.3), all the values of the timePropertiesDS data set (see 8.2.4), and timing signals from the Local PTP Clock.

The external interface of the Sink Adapter block is specific to the Clock Sink.

NOTE—Examples of this mechanism include a 10 MHz signal, PPS signal, and data link, or direct read access to the Local PTP Clock of the sourcing PTP Instance.

7.6.7.3 External Sink block

The External Sink block provides timing information and signals to one or more Sink Adapter blocks.

NOTE—In Figure 5 and Figure 6, each arrow pointing to the Sink Adapter Blocks points to a different Sink Adapter block.

7.7 PTP timing characterization

7.7.1 General

Subclause 7.7 specifies the timing and timeout attributes of the protocol.

7.7.2 PTP message transmission intervals

7.7.2.1 General interval specification

For each of the PTP message types Announce, Sync, Delay_Req, and Pdelay_Req, the mean time interval between successive PTP messages shall be represented as the logarithm to the base 2 of this time interval measured in seconds. The values of these logarithmic attributes shall be selected from integers in the range –128 to 127 subject to further limits established in the applicable PTP Profile. These intervals are communicated via the logMessageInterval field of PTP messages. The interpretation of the logMessageInterval depends on the message type (see 13.3.2.14).

7.7.2.2 Announce message transmission interval

The portDS.logAnnounceInterval shall specify the mean time interval between successive Announce messages, that is, the announceInterval (see 9.5.8).

The portDS.logAnnounceInterval should be uniform throughout a domain. The behavior of domains in which this is not so is outside the scope of this standard.

NOTE 1—The value of portDS.logAnnounceInterval is a compromise between the desired responsiveness to changes in the PTP Network in determining the master–slave hierarchy in a domain and the communication and computation load imposed by transmission of these messages.

NOTE 2—It might be desirable for the portDS.logAnnounceInterval to be different in regions of different communication technologies, for example, wired and wireless technologies. Systems where the portDS.logAnnounceInterval varies from region to region need to adjust the value of the portDS.announceReceiptTimeout on all PTP Ports to ensure that timeouts do not occur as a result of the differences in the announceIntervals. Even with this precaution, reconfiguration of the entire PTP Network will usually take longer than with comparable uniform intervals.

7.7.2.3 Sync (multicast) message transmission interval

The portDS.logSyncInterval shall specify the mean time interval between successive Sync messages, that is, the syncInterval, when transmitted as multicast messages (see 9.5.9.2).

NOTE 1—It might be desirable for the portDS.logSyncInterval to be different in regions of different communication technologies, for example, wired and wireless technologies.

NOTE 2—The value of portDS.logSyncInterval is a compromise between the stability and precision of the Local Clocks and the communication and computation load imposed by transmission of these messages.

7.7.2.4 Delay_Req message transmission interval

The `portDS.logMinDelayReqInterval` shall specify the minimum permitted mean time interval between successive Delay_Req messages, that is, the `minDelayRequestInterval`, sent by a Slave PTP Instance to a specific PTP Port on the Master PTP Instance (see 9.5.11.2).

This value is determined and advertised by a Master PTP Instance based on the ability of the Master PTP Instance to process the Delay_Req message traffic. The minimum and maximum values shall be integer values specified in the applicable PTP Profile.

NOTE—The value of `portDS.logDelayReqInterval` is a compromise between the responsiveness in changes to path delay and the communication and computation load imposed by transmission of these messages.

7.7.2.5 Pdelay_Req message transmission interval

The `portDS.logMinPdelayReqInterval` shall specify the minimum permitted mean time interval between successive Pdelay_Req messages, that is, the `minPdelayReqInterval`, sent over a PTP Link (see 9.5.13.2).

NOTE—The value of `portDS.logMinPdelayReqInterval` is a compromise between the fluctuation in PTP Link delay and startup time and the communication and computation load imposed by transmission of these messages.

7.7.3 PTP timeouts

7.7.3.1 `portDS.announceReceiptTimeout`

The value of `portDS.announceReceiptTimeout` shall specify the number of `announceIntervals` that must pass without receipt of an Announce message before the occurrence of the event `ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES` (see 9.2.6.12). The range shall be 2 to 255 subject to further restrictions of the applicable PTP Profile. Although 2 is permissible, normally the value should be at least 3.

NOTE 1—A value of 2 is permitted but is not recommended except in carefully controlled environments where more rapid reaction to missed Announce messages is needed and the number of missed messages is very low.

The product $(\text{portDS.announceReceiptTimeout}) \times 2^{\text{portDS.logAnnounceInterval}}$ should be uniform throughout a domain. The behavior of domains for which this is not the case is outside the scope of this standard.

8. PTP data sets

8.1 General specifications for data set members

8.1.1 Introduction to data sets

8.1.1.1 Introduction to data set specifications

Each data set member specification includes:

- The formal name for the member
- A reference or definition of the semantics associated with the member
- Initialization and configuration properties (see 8.1.3)

The specifications of data set members in 8.1 shall also apply to the optional data set members defined by optional features of this standard unless these features specify otherwise.

8.1.1.2 Ordinary Clocks and Boundary Clocks

For each Ordinary Clock and Boundary Clock, the following PTP Instance data sets shall be maintained locally as the basis for protocol decisions and computations, and for providing values for PTP message fields:

- defaultDS (see 8.2.1)
- currentDS (see 8.2.2)
- parentDS (see 8.2.3)
- timePropertiesDS (see 8.2.4)
- portDS (one data set for each PTP Port of an Ordinary Clock or Boundary Clock; see 8.2.15)

For each Ordinary Clock and Boundary Clock, the PTP Instance data sets applicable to optional features of the protocol shall, if the option is implemented, be maintained locally as the basis for protocol decisions and computations and for providing values for PTP message fields.

8.1.1.3 Transparent Clocks

The 2008 edition of this standard specified the following data sets for each “Transparent Clock”:

- transparentClockDefaultDS data set (see 8.3.2)
- transparentClockPortDS data set (one data set for each PTP Port; see 8.3.3)

According to the principles specified in 7.1, each data set is scoped to a single domain and, therefore, to a single PTP Instance. In the 2008 edition, although it was not clearly specified, the data sets for Transparent Clocks were typically interpreted as applicable to the PTP Node. When a PTP Node contains a single PTP Instance, this interpretation aligned with the design principles of 7.1. When a PTP Node contains more than one PTP Instance, this interpretation has limitations that are not acceptable as the standard progresses.

To resolve this issue, this edition of the standard specifies transparentClockDefaultDS and transparentClockPortDS as applying to the PTP Node, but it designates those data sets as deprecated (4.2.8). This edition of the standard provides a methodology for Transparent Clock data sets for each PTP Instance, and that methodology is recommended for new designs (see 8.3.1).

8.1.2 Initialization classification

8.1.2.1 General

Every member of a data set is classified as “static,” “dynamic,” or “configurable.”

8.1.2.1.1 Static members

The values of static members are inherent physical or operational properties of the PTP Instance or of the protocol.

8.1.2.1.2 Dynamic members

The values of dynamic members are not directly changed by users, but they may change as follows:

- As a result of protocol operations. For example, the portDS.portState may change as a result of protocol events.
- Due to changes in the internal properties of the PTP Instance. For example, the defaultDS.clockQuality.offsetScaledLogVariance may change due to temperature effects on the local oscillator.
- Due to interactions with timing systems external to PTP. For example, a PTP Instance that can synchronize its Local PTP Clock to the GPS system may change the value of its defaultDS.clockQuality, defaultDS.currentTime, or timePropertiesDS data set members when it first locks to GPS.

8.1.2.1.3 Configurable members

The values of configurable members can only be changed using management mechanisms or implementation-specific configuration means.

Unless otherwise stated in this standard, when the value of a configurable member is updated, the updated value shall take effect immediately upon update, consistent with atomicity requirements (e.g., 9.6).

Unless otherwise stated in this standard, the updated values for configurable members shall be restricted in range to the most restrictive of the range values specified in Clause 7 and of the applicable PTP Profile.

NOTE 1—For example, the value of the defaultDS.domainNumber can be changed via a management mechanism but is otherwise unaffected by the protocol or internal changes in the PTP Instance.

NOTE 2—Due to the requirements of 4.2.7, if a member is configured to a value that is specified as reserved in this standard, the PTP Instance is not allowed to change to that reserved value. Ideally, the PTP Instance will return an error in the management mechanism to indicate that the configuration failed.

8.1.3 PTP Instance data set initialization properties

8.1.3.1 General initialization specifications

The initialization properties of a data set member shall be determined by its classifications as “static,” “dynamic,” or “configurable.”

For an Ordinary Clock or a Boundary Clock, the initialization actions indicated in the INITIALIZING row of Table 27, PTP portState definition, shall be completed before leaving the INITIALIZATION state.

For a Transparent Clock, data set members shall be initialized before beginning normal operation as specified in 8.1.3.2, 8.1.3.3, and 8.1.3.4.

8.1.3.2 Initialization of static data set members

Static members shall be initialized to the implementation-specific value meeting the specifications for the member.

8.1.3.3 Initialization of dynamic data set members

Dynamic members shall be initialized to the first of the following values that applies:

- a) The value mandated in the specification for the data set member
- b) The value that represents the properties of the PTP Instance or protocol at the time of initialization

NOTE 1—For example, the defaultDS.clockQuality value can depend on whether the Local PTP Clock is synchronized to GPS at initialization.

- c) The value in nonvolatile read–write storage, if implemented

NOTE 2—For example, an implementation can store the value of currentDS.meanDelay in nonvolatile storage on the assumption that the PTP Network is unlikely to be reconfigured frequently.

- d) Implementation-specific value

If dynamic member values are maintained in nonvolatile read–write memory, the manufacturer should preload this memory with the applicable value from item a) or item b) or item d).

The values from item a) or item b) or item d) are called the “dynamic data set initialization values.”

8.1.3.4 Initialization of configurable data set members

Configurable members shall be initialized to the first one of the following values that applies:

- a) Last value configured by management mechanisms or implementation-specific means and maintained in nonvolatile read–write storage, if implemented
- b) The default initialization value for the member as specified in the applicable PTP Profile implemented by the PTP Instance
- c) The initialization value indicated in the member specifications

If the configured values are maintained in nonvolatile read–write memory, the manufacturer should preload this memory with the applicable initialization value from item b) or item c).

The values from item b) or item c) are called the “configurable data set initialization values.”

8.1.3.5 Operation of nonvolatile read–write storage of data set members

The current values of one or more dynamic or configurable data set members may be maintained in nonvolatile read–write storage.

NOTE—For example, this can allow for more rapid configuration of systems after a power outage.

The contents of nonvolatile read–write memory shall be reset to the applicable dynamic or configurable data set initialization values (see 8.1.3.3 and 8.1.3.4) when the data set member `nonVolatileStorageDS.reset` is written to TRUE or when the implementation-specific equivalent is performed.

The current values of the applicable dynamic and configurable data set members shall be copied into nonvolatile read–write memory when the data set member `nonVolatileStorageDS.save` is written to TRUE or when the implementation-specific equivalent is performed.

Some management mechanisms specify their own requirements for handling nonvolatile storage (e.g., NETCONF [B20]). These requirements shall take precedence, and the `nonVolatileStorageDS.reset` and `nonVolatileStorageDS.save` members shall not be supported if they contradict the requirements of such a management mechanism.

8.1.4 Use of data sets for management

8.1.4.1 General

Although Clause 15 specifies PTP management messages, there are other standards for remote management of network infrastructure, and PTP Nodes are considered to be network infrastructure.

Management standards are often based on a standard data modeling language, such as SMIV2 (IETF STD 58 [B26], commonly known as “MIB”) or YANG (IETF RFC 7950 [B22]). The data modeling language is used to create a textual file (module) that provides a formal specification for the data to be managed. Standard management protocols build on the data modeling language to specify on-the-wire formats and other specifics such as the transport protocol and security. For example, the SNMP protocol (IETF STD 62 [B27]) uses MIB standards, and the NETCONF protocol (IETF RFC 6241 [B20]) uses YANG standards.

To use a data modeling language to create a textual file (module), a formal specification for the data is needed, called the “information model.” The information model specifies the hierarchy of data items within the device (which for this standard is the PTP Node). For each item of data, the information model specifies the item’s semantics (such as the description and data type), conformance, and read/write permissions.

Although the primary use of PTP data sets is the operation of the PTP protocol and state machines, the PTP data sets also serve as the information model for management.

The specifications of the data sets can be mapped to specific data models, including MIB and YANG. The PTP management messages of Clause 15 represent another mapping of the data sets for management purposes.

8.1.4.2 Organization within a PTP Node

Management data models typically represent data for the physical device (i.e., PTP Node). The specifications for discovery, management address, and security for the physical device are typically covered by standards of the management mechanism, which are outside the scope of this standard. For the management information model of this standard, the scope of work is the data contained within a PTP Node.

From a management perspective, the PTP Node contains a list of one or more PTP Instances. Each entry in the list provides a `defaultDS.instanceType` member (8.2.1.5.5) that specifies the type of PTP Instance (i.e., Ordinary Clock, Boundary Clock, end-to-end Transparent Clock, or peer-to-peer Transparent Clock).

The following hierarchy summarizes the managed data sets within a PTP Node:

- a) `instanceList[]`
 - 1) `defaultDS`
 - 2) `currentDS`
 - 3) `parentDS`
 - 4) `timePropertiesDS`
 - 5) *`descriptionDS`*
 - 6) *`faultLogDS`*
 - 7) *`nonVolatileStorageDS`*
 - 8) *`pathTraceDS`*
 - 9) *`alternateTimescaleOffsetsDS`*
 - 10) *`holdoverUpgradeDS`*
 - 11) *`grandmasterClusterDS`*
 - 12) *`acceptableMasterTableDS`*
 - 13) *`performanceMonitoringDS`*
 - 14) *`enhancedSynchronizationAccuracyMetricsDS`*
 - 15) `portList[]`
 - i) `portDS`
 - ii) *`timestampCorrectionPortDS`*
 - iii) *`asymmetryCorrectionPortDS`*
 - iv) *`descriptionPortDS`*
 - v) *`unicastNegotiationPortDS`*
 - vi) *`alternateMasterPortDS`*
 - vii) *`unicastDiscoveryPortDS`*
 - viii) *`acceptableMasterPortDS`*
 - ix) *`LISyncBasicPortDS`*
 - x) *`LISyncOptParamsPortDS`*
 - xi) *`communicationCapabilitiesPortDS`*
 - xii) *`performanceMonitoringPortDS`*
 - xiii) *`slaveMonitoringPortDS`*
 - xiv) *`commonServicesPortDS`*
- b) *`transparentClockDefaultDS`*
- c) *`transparentClockPortList[]`*
 - 1) *`transparentClockPortDS`*

- d) *commonServices*
 - 1) *commonMeanLinkDelayService*
 - i) *cmlDsDefaultDS*
 - ii) *cmlDsLinkPortList[]*
 - i) *cmlDsLinkPortDS*
 - ii) *cmlDsTimestampCorrectionLinkPortDS*
 - iii) *cmlDsAsymmetryCorrectionLinkPortDS*
 - iv) *cmlDsPerformanceMonitoringLinkPortDS*

In this hierarchy, the names of optional data sets are italicized. It is intended only as a summary. If any contradiction becomes apparent between the normative text of this standard and this hierarchy, the normative text takes precedence.

The deprecated Transparent Clock data sets and the common service data sets are provided at the top level of the hierarchy, applicable to the entire PTP Node (see 8.1.1.3).

The instanceList is indexed using a number that is unique per PTP Instance within the PTP Node, applicable to the management context only (i.e., not used in PTP messages). The domainNumber of the PTP Instance must not be used as the index to instanceList since it is possible for a PTP Node to contain multiple PTP Instances using the same domainNumber.

PTP Instances and PTP Ports may be created or deleted dynamically in implementations that support dynamic create/delete of devices.

NOTE 1—This hierarchy is intended to support a wide variety of PTP Node implementations. Some examples include a) PTP Node containing four Boundary Clocks, each of which use the same physical ports, but different domainNumber values; b) PTP Node with two PTP Instances, an Ordinary Clock and Transparent Clock; and c) PTP Node that represents a chassis with slots for switch/router cards; each switch/router card is represented as a PTP Instance using distinct physical ports; all PTP Instances can use the same domainNumber.

NOTE 2—The preceding hierarchy is an information model that is intended to guide specifications for a management mechanism capable of managing any PTP Node permitted by the standard. However, if for example, a PTP Profile supports only one PTP Instance, its management mechanism (e.g., MIB) can omit the instanceList (i.e., the defaultDS will be at the top level).

8.1.4.3 Management conformance

Unless management is explicitly specified, conformance statements (e.g., “shall” or “may”) for data set members refer to usage of the data set member for operation of the PTP protocol and its state machines (see 8.1.6).

A PTP Node may support management of one or more data set members. If management of data members is supported, one of the following techniques shall be used:

- One or more management mechanisms specified in the applicable PTP Profile or in product specifications. Each management mechanism shall specify a data model that conforms to the data set specifications of this standard. For example, the PTP management messages of 15.2 are a conformant protocol, and the TLVs of those messages are a conformant data model.
- The applicable PTP Profile or the product specification shall specify the defined fixed values and/or state that there is an implementation-specific means to address all configurable members (see 8.1.2.1.3).

If management of data members is not supported, one of the following techniques shall be used:

- The applicable PTP Profile or the product specification shall specify defined fixed values for all configurable members (see 8.1.2.1.3), or
- The values for all configurable members (see 8.1.2.1.3) shall be the default initialization values specified for the configurable members (see 8.1.3.4)

8.1.4.4 Read/write permissions

Many data modeling languages provide a mechanism to specify read/write permissions from the client's perspective. A read-only attribute is sometimes referred to as “state” data, and a read-write attribute is sometimes referred to as “configuration” data.

Static data set members shall be modeled as read-only for management.

Configurable data set members shall be modeled as read-write for management unless otherwise specified in the standard.

8.1.5 Data type

For each data set member, a data type is specified as mandatory (i.e., “shall”). There are two aspects to specification of a data type, as follows:

- Semantic: Meaning of the data type (e.g., `UInteger16` represents numbers from 0 to 65535)
- Syntax: How the data is formatted (e.g., `UInteger16` is two octets with most significant octet first)

For products to interoperate on the wire, both semantic and syntax are required for each data type. Nevertheless, separation of the semantic and syntax requirement can provide flexibility for a variety of implementations and on-the-wire protocols.

The data type requirement for the data sets of this subclause shall be interpreted as semantic only (not as syntax).

This provides flexibility for implementations, which can use whatever syntax is appropriate. For example, an ANSI C implementation can use “unsigned short” for this standard's `UInteger16`, and the encoding can be two octets with least significant octet first (assuming this is the processor's byte order).

This provides flexibility for on-the-wire protocols, each of which will complete the specification of the data set member with a syntax requirement, as well as any semantic clarifications that are needed.

One clear example of an on-the-wire protocol is the use of data set members within PTP messages. The syntax for each data type uses the rules of 5.4, as well additional specifications for the PTP message.

Another example of on-the-wire protocols is management mechanisms such as SNMP [B27], NETCONF [B20], and RESTCONF [B23]. Some of these management mechanisms use textual syntax on the wire, such that a data type like `UInteger16` is formatted as a string from “0” to “65535”. To specify a mapping of this standard's data types to the analogous data type of the management mechanism, the data sets of this standard specify the semantic data type of each member.

8.1.6 Operational conformance

Each data set is introduced with a table that summarizes the operational conformance of each member. Conformance refers to statements that specify what the standard permits (e.g., “shall” or “may”). Operational refers to usage of data set members for operation of the PTP protocol and its state machines, and for values of message fields.

Operational conformance does not refer to management access to data set members, which is optional (see 8.1.4.3).

The operational conformance table for each data set is intended as a summary only. If any contradiction becomes apparent between the normative text of the standard and the operational conformance table, the normative text takes precedence.

The columns of the operational conformance table are:

- **Name:** The name of the data set member.
If the name refers to a subordinate data set, the conformance applies to the subordinate data set as a whole, and if supported, the subordinate data set provides another operational conformance table for each of its data set members.
- **OC:** Operational conformance for an Ordinary Clock.
- **BC:** Operational conformance for a Boundary Clock.
- **E2E TC:** Operational conformance for an end-to-end Transparent Clock (10.2).
- **P2P TC:** Operational conformance for a peer-to-peer Transparent Clock (10.3).

For both E2E TC and P2P TC, the columns refer to operational conformance when the Transparent Clock supports PTP Instance data sets (8.3.1.2, 8.2). As an alternative, the Transparent Clock can support PTP Node data sets (8.3.1.1, 8.3.2, 8.3.3), for which operational conformance is deprecated.

The conformance terms used in each cell of the table are:

- **required:** The data set member is a mandatory requirement of this standard (4.2.1).
- **optional:** The data set member is optional for this standard (4.2.3, 4.2.4).
A PTP Profile can require the optional data set member.
As specified in 4.2.6, an optional feature can have mandatory requirements when it is supported.
- **deprecated:** The data set member is deprecated for this standard (4.2.8).
- **N/A:** The data set member is not applicable.
- **management:** The data set member has no operational conformance and is used exclusively for management purposes (8.1.4.3).

8.1.7 Correspondence between abstract quantities and data set members

For any quantity defined in this standard using the angle bracket, $\langle \rangle$, notation of 4.1.1, that has a corresponding data set member:

- If the quantity identified by angle brackets is measured or evaluated, then the corresponding data set member shall be updated to reflect the measured or evaluated value.
- If a data set member for which there is a corresponding quantity defined by angle brackets is updated by the operation of the protocol or by a management mechanism, then the corresponding quantity defined by angle brackets shall be updated to reflect the new value of the data set member.

NOTE—An example of such a correspondence is $\langle \text{delayAsymmetry} \rangle$ and portDS.delayAsymmetry.

8.2 Data sets for PTP Instances

The following data sets apply to a single PTP Instance. An Ordinary Clock or Boundary Clock is always represented by these data sets. The data sets of this subclause should be used to represent Transparent Clocks.

NOTE—See 8.3.1 for the exception allowing a PTP Node level data set to be used in certain circumstances.

8.2.1 defaultDS data set member specifications

8.2.1.1 General

The name and operational conformance, as defined in 8.1.6, of each defaultDS member is summarized in Table 7. Members listed as management are used only by management mechanisms for purposes of configuration and monitoring.

Table 7—defaultDS operational conformance

Name	OC	BC	E2E TC	P2P TC
twoStepFlag	deprecated	deprecated	N/A	N/A
clockIdentity	required	required	optional ¹³	required
numberPorts	required	required	optional	required
clockQuality	required	required	N/A	N/A
priority1	required	required	N/A	N/A
priority2	required	required	N/A	N/A
domainNumber	required	required	required	required
slaveOnly	required	N/A	N/A	N/A
currentTime	management	management	management	management
instanceEnable	optional	optional	N/A	N/A
externalPortConfigurationEnabled	optional	optional	N/A	N/A
maxStepsRemoved	optional	optional	N/A	N/A
sdoId	required	required	required	required
instanceType	management	management	management	management

8.2.1.2 Static members of the defaultDS data set

8.2.1.2.1 defaultDS.twoStepFlag (deprecated)

The value of defaultDS.twoStepFlag shall be TRUE if the PTP Instance is a two-step PTP Instance; otherwise, the value shall be FALSE. The data type shall be Boolean.

NOTE 1—One-step or two-step egress behavior is allowed to be specified per PTP Port, or per PTP Instance. Management of the one/two-step egress behavior of a PTP Port is not provided by this standard but can be specified as extensions to the data sets by a PTP Profile or a product specification.

NOTE 2—This data set member is no longer used (see 7.5.2.5). However, the twoStepFlag of the PTP common header is used; see Table 37 of 13.3.2.8.

¹³If an E2E TC supports an optional feature that initiates transmit of a PTP message (e.g., PTP management messages), data set members that are used to form sourcePortIdentity are required (e.g., defaultDS.clockIdentity).

8.2.1.2.2 defaultDS.clockIdentity

The value of defaultDS.clockIdentity shall be the value of clockIdentity (see 7.6.2.2) of the PTP Instance. The data type shall be ClockIdentity.

8.2.1.2.3 defaultDS.numberPorts

The value of defaultDS.numberPorts shall be the number of PTP Ports on the PTP Instance. For an Ordinary Clock, this value shall be 1. The data type shall be UInteger16.

8.2.1.3 Dynamic members of the defaultDS data set

8.2.1.3.1 defaultDS.clockQuality

8.2.1.3.1.1 General

The data type of this member shall be ClockQuality (see 5.3.7).

8.2.1.3.1.2 defaultDS.clockQuality.clockClass

The value of defaultDS.clockQuality.clockClass shall follow the clockClass specifications of 7.6.2.5.

The initialization value of defaultDS.clockQuality.clockClass shall be selected as follows:

- The value is dependent on the initialization value of the defaultDS data set member defaultDS.slaveOnly (see 8.2.1.4.4), which shall be initialized prior to initialization of the defaultDS.clockQuality.clockClass member.
- If defaultDS.slaveOnly is TRUE, the initialization value shall be 255 as specified in 7.6.2.5.
- If defaultDS.slaveOnly is FALSE and the applicable PTP Profile specifies the clockClass to be 52, 58, 187, 193, or in the ranges 68 through 122, 133 through 170, or 216 through 232, then the PTP Profile specified clockClass value shall be used for initialization.
- If defaultDS.slaveOnly is FALSE and if the PTP Instance is designed as a clockClass 6 or 13, the clockClass initialization value shall be 6 or 13, respectively, if these represent the clockClass of the PTP Instance upon exiting the INITIALIZING state. If the clockClass 6 or 13, respectively, does not represent the PTP Instance upon exiting the INITIALIZING state, the clockClass initialization value shall be as follows:
 - 1) Either 52, 187, or 248, as specified in the applicable PTP Profile, for a PTP Instance designed as class 6.
 - 2) Either 58, 193, or 248, as specified in the applicable PTP Profile, for a PTP Instance designed as class 13.
- Else, the value shall be 248.

NOTE—A PTP Instance that is designed to be clockClass 6 or 13 can include implementation-specific measures to ensure it meets the specifications of 7.6.2.5 for these clockClasses before initializing the defaultDS.clockQuality.clockClass member. For example, the PTP Instance can synchronize to the GPS system as part of the activities in the INITIALIZING state prior to initializing the defaultDS.clockClass member. If such a PTP Instance is unable to meet the specifications of 7.6.2.5 prior to leaving the INITIALIZING state, then one of the degradation or the default alternatives for the PTP Instance is used.

8.2.1.3.1.3 defaultDS.clockQuality.clockAccuracy

The value of defaultDS.clockQuality.clockAccuracy shall follow the clockAccuracy specifications of 7.6.2.6.

The initialization value of defaultDS.clockQuality.clockAccuracy shall be selected as follows:

- a) The value is dependent on the initialization value of the defaultDS.clockQuality.clockClass (see 8.2.1.3.1.2), which shall be initialized prior to initialization of clockAccuracy.
- b) The clockAccuracy initialization value shall represent the clockAccuracy of the PTP Instance at the time of initialization as specified in 7.6.2.6.

8.2.1.3.1.4 defaultDS.clockQuality.offsetScaledLogVariance

The value of defaultDS.clockQuality.offsetScaledLogVariance shall follow the offsetScaledLogVariance specifications of 7.6.2.7.

The initialization value of the defaultDS.clockQuality.offsetScaledLogVariance shall reflect the inherent characteristics of the PTP Instance at the time of initialization as specified in 7.6.2.7.

8.2.1.4 Configurable members of the defaultDS data set

8.2.1.4.1 defaultDS.priority1

The value of defaultDS.priority1 is the value of the priority1 attribute (see 7.6.2.3) of the PTP Instance. The data type shall be UInteger8.

8.2.1.4.2 defaultDS.priority2

The value of defaultDS.priority2 is the value of the priority2 attribute (see 7.6.2.4) of the PTP Instance. The data type shall be UInteger8.

8.2.1.4.3 defaultDS.domainNumber

The value of defaultDS.domainNumber is the value of the domainNumber attribute (see 7.1) of the PTP Instance. The data type shall be UInteger8.

8.2.1.4.4 defaultDS.slaveOnly

The value of defaultDS.slaveOnly shall be TRUE if the PTP Instance is a slave-only PTP Instance (see 9.2.2.1). The value shall be FALSE if the PTP Instance is a non-slave-only PTP Instance. The data type shall be Boolean.

8.2.1.4.5 defaultDS.sdoId

The value of the defaultDS.sdoId is the value of the sdoId attribute (see 7.1 and 7.1.4) of the domain. The value of the defaultDS.sdoId member shall be selected based on Table 2. The specification initialization value (see 8.1.3.4) shall be 000₁₆.

The data type shall be UInteger12.

8.2.1.5 Optional members of the defaultDS data set

For the following optional members, the classification (i.e., static, dynamic, configuration) is provided in each member's specification.

8.2.1.5.1 defaultDS.currentTime

This member may be supported for management (see 8.1.6).

The data type shall be Timestamp. The classification is configurable.

For management read, this member shall return the current value of the PTP Instance Time (3.1.54).

When management write is supported, this member shall set the PTP Instance Time.

Time originates in the Grandmaster PTP Instance and is distributed by PTP to other PTP Instances in the domain.

NOTE 1—The time in the Grandmaster PTP Instance is normally determined by interacting with a primary reference, for example, GPS, by means outside the scope of this standard.

NOTE 2—When this member is used to set time in a PTP Instance other than the Grandmaster PTP Instance, the PTP Node can return a management error.

NOTE 3—If the time is set in a PTP Instance other than the Grandmaster PTP Instance, it will be overwritten by the operation of the protocol and will therefore exist only as a transient.

NOTE 4—Since this member's value is transmitted to the PTP Instance using management, the accuracy to which the time can be set is limited by network fluctuations. For this reason, no provision is made for setting the time to a precision of fractions of a nanosecond. In most cases, the actual precision is on the order of milliseconds or worse depending on the source of the information used in populating the data field and on the characteristics of the network.

8.2.1.5.2 defaultDS.instanceEnable

For management read, this member indicates if the PTP Instance is enabled for PTP operation.

When management write is supported:

- Write of the value TRUE shall cause the INITIALIZE event (9.2.6.3) to occur, only if the value was previously FALSE.
- Write of the value FALSE shall immediately disable operation of the PTP Instance (i.e., analogous to power off).

The data type shall be Boolean. The classification is configurable.

If this data set member is not supported, the PTP Instance shall be specified-by-design to be enabled.

Unless stated otherwise by product specifications, the specification initialization value (see 8.1.3.4) shall be TRUE.

8.2.1.5.3 defaultDS.externalPortConfigurationEnabled

The defaultDS.externalPortConfigurationEnabled is defined in 17.6.2. Its value indicates whether the external port configuration option of 17.6 is in the disabled or enabled state.

8.2.1.5.4 defaultDS.maxStepsRemoved

The value of defaultDS.maxStepsRemoved is used by the option specified in item e) of 9.3.2.5. When using this option, if the value of stepsRemoved of an Announce message is greater than or equal to the value of defaultDS.maxStepsRemoved, the Announce message is not considered in the operation of the BMCA specified in 9.3.2, 9.3.3, and 9.3.4. The value shall be in the closed range 2 to 255. The default value shall be 255. If the option is inactive (not used), the value shall be 255.

The data type shall be UInteger8. The classification is configurable.

8.2.1.5.5 defaultDS.instanceType

This member may be supported for management (see 8.1.6).

The classification of defaultDS.instanceType is configurable unless the value is set-by-design, in which case it is static. The defaultDS.instanceType member is read-only unless support for write is explicitly specified by the applicable PTP Profile or product specification.

NOTE 1—For example, if a product supports a single PTP Instance that can be configured as either Boundary Clock or end-to-end Transparent Clock, it can support this with a writable instanceType.

The data type for instanceType shall be Enumeration8, using an enumerated value from Table 8.

Table 8—instanceType enumeration

Value (hex)	Description
00	Ordinary Clock (OC)
01	Boundary Clock (BC)
02	peer-to-peer Transparent Clock (TC)
03	end-to-end Transparent Clock (TC)
04-FF	Reserved

For management read, this member indicates the type of PTP Instance.

For management write, if the type in the write is supported, the type is changed and the INITIALIZE event (9.2.6.3) shall occur.

NOTE 2—This instanceType is similar in purpose to the clockType of PTP management messages (15.5.3.1.2.1). For backward compatibility, PTP management messages continue to use clockType (not instanceType).

8.2.2 currentDS data set member specifications

8.2.2.1 General

The name and operational conformance, as defined in 8.1.6, of each currentDS member is summarized in Table 9.

Table 9—currentDS operational conformance

Name	OC	BC	E2E TC	P2P TC
stepsRemoved	required	required	N/A	N/A
offsetFromMaster	required	required	N/A	N/A
meanDelay	required	required	N/A	N/A
synchronizationUncertain	optional	optional	N/A	N/A

All members of the currentDS data set are dynamic.

8.2.2.2 currentDS.stepsRemoved

The value of currentDS.stepsRemoved is the number of PTP Communication Paths traversed between the PTP Instance and the Grandmaster PTP Instance.

The data type shall be UInteger16. The initialization value shall be 0.

NOTE—For example, currentDS.stepsRemoved for a Slave PTP Instance on the same PTP Communication Path as the Grandmaster PTP Instance has a value of 1, indicating that a single path was traversed.

8.2.2.3 currentDS.offsetFromMaster

The value of currentDS.offsetFromMaster shall be the current value of the time difference between a Master PTP Instance and a Slave PTP Instance as computed by the Slave PTP Instance; that is, $\langle \text{offsetFromMaster} \rangle = \langle \text{Time on the Slave PTP Instance} \rangle - \langle \text{Time on the Master Clock} \rangle$ (see 11.2).

The data type shall be TimeInterval. The initialization value shall be either:

- The value in nonvolatile read-write storage, if implemented
- Implementation specific

8.2.2.4 currentDS.meanDelay

The value of the currentDS.meanDelay shall be 0 when the PTP Port that is in the SLAVE or UNCALIBRATED state is also configured SPECIAL or NO_MECHANISM using the portDS.delayMechanism (see Table 21 of 8.2.15.4.4). Otherwise, the value of the currentDS.meanDelay is as follows:

- Zero when none of the PTP Ports in the PTP Instance is in the SLAVE or UNCALIBRATED state (see 9.3.5 and 17.6.5.4), otherwise
- The current value calculated on the PTP Port in the SLAVE or UNCALIBRATED state of the mean propagation time

- Over the PTP Communication Path if the delay request-response mechanism is in use, that is, `<meanPathDelay>` (see 11.3), or
- Over the preceding PTP Link connected to the PTP Port in the SLAVE or UNCALIBRATED state, if the peer-to-peer delay mechanism is in use, that is, `<meanLinkDelay>` (see 11.4)

The data type should be `TimeInterval`. The initialization value shall be either:

- The value in nonvolatile read–write storage, if implemented, or
- Implementation specific

NOTE 1—See NOTES in 7.4.2.

NOTE 2—In the 2008 edition of this standard, this data set member was called “`currentDS.meanPathDelay`.” Although the specification of this member is retained in the current edition of this standard, the member is renamed to “`currentDS.meanDelay`.” This change is consistent with other changes that ensure clarity and consistency of naming, where

- “path” is associated with the delay request-response mechanism, for example, `<meanPathDelay>`, PTP Communication Path
- “link” is associated with the peer-to-peer delay mechanism, for example, `<meanLinkDelay>`, PTP Link

This member is associated with both delay request-response and peer-to-peer mechanism; thus, neither “path” nor “link” is used in its name.

8.2.2.5 `currentDS.synchronizationUncertain`

This data set member is optional. If the applicable PTP Profile specifies use of the `synchronizationUncertain` flag in Announce messages, this data set member shall be `FALSE` except under any of the following conditions for which it shall be `TRUE`:

- a) The `synchronizationUncertain` flag of the Announce message received from the Parent PTP Instance (`parentDS.synchronizationUncertain`) is `TRUE`, or
- b) The ingress PTP Port is in the UNCALIBRATED state, or
- c) The applicable PTP Profile has specified a performance metric defining when the PTP Instance Time is within specification, and the PTP Instance Time fails to meet this performance metric. If the applicable PTP Profile does not specify a performance metric, then only condition a) and condition b) are applicable. If the applicable PTP Profile does not specify the use of this flag by the Grandmaster, then the Grandmaster sets this flag to `FALSE`.

Unless otherwise specified in the applicable PTP Profile, the interpretation of the `synchronizationUncertain` flag by the PTP Instance receiving the Announce message, except for the provisions above, is implementation specific.

If `currentDS.synchronizationUncertain` is supported, its value shall be used as the `synchronizationUncertain` flag of Announce messages transmitted from an egress PTP Port.

The data type shall be `Boolean`. The specification initialization value shall be `FALSE`.

8.2.3 parentDS data set member specifications

8.2.3.1 General

The name and operational conformance, as defined in 8.1.6, of each parentDS member is summarized in Table 10.

Table 10—parentDS operational conformance

Name	OC	BC	E2E TC	P2P TC
parentPortIdentity	required	required	N/A	N/A
parentStats	required	required	N/A	N/A
observedParentOffsetScaledLogVariance	optional	optional	N/A	N/A
observedParentClockPhaseChangeRate	optional	optional	N/A	N/A
grandmasterIdentity	required	required	N/A	N/A
grandmasterClockQuality	required	required	N/A	N/A
grandmasterPriority1	required	required	N/A	N/A
grandmasterPriority2	required	required	N/A	N/A
protocolAddress	optional	optional	N/A	N/A
synchronizationUncertain	optional	optional	N/A	N/A

The parentDS data set shall be initialized after the defaultDS data set is initialized.

All members of the parentDS data set are dynamic.

8.2.3.2 parentDS.parentPortIdentity

The data type shall be PortIdentity (see 5.3.5). The value of parentDS.parentPortIdentity is the value of the portIdentity of the PTP Port on the Master PTP Instance that issues the Sync messages used in synchronizing the Local PTP Clock of this PTP Instance.

The initialization value shall be as follows:

- The value of the parentDS.parentPortIdentity.clockIdentity member is the value of the defaultDS.clockIdentity field.
- The parentDS.portNumber member is 0.

The data type shall be PortIdentity (5.3.5).

8.2.3.3 parentDS.parentStats

The value of parentDS.parentStats shall be TRUE if all of the following conditions are satisfied:

- The PTP Instance has a PTP Port in the SLAVE state.
- The PTP Instance has computed statistically valid estimates of the parentDS.observedParentOffsetScaledLogVariance and the parentDS.observedParentClockPhaseChangeRate members.

Otherwise the value shall be FALSE.

The data type shall be Boolean. The initialization value shall be FALSE.

8.2.3.4 parentDS.observedParentOffsetScaledLogVariance

The value of parentDS.observedParentOffsetScaledLogVariance of the Local PTP Clock shall be an estimate of the variance of the phase offset of the Local PTP Clock of the Parent PTP Instance as measured with respect to the Local PTP Clock in the Slave PTP Instance, measured and represented as described in 7.6.3.3 and 7.6.3.5. This measurement is optional, but if not made, the value of parentDS.parentStats shall be FALSE.

The data type shall be UInteger16. The initialization value shall be FFFF₁₆ (see 7.6.3.3).

8.2.3.5 parentDS.observedParentClockPhaseChangeRate

The value of parentDS.observedParentClockPhaseChangeRate shall be an estimate of the phase change rate of the Local PTP Clock of the Parent PTP Instance as measured by the Slave PTP Instance using its Local PTP Clock as defined in 7.6.4.4. If the estimate exceeds the capacity of its data type, this value shall be set to 7FFF FFFF₁₆ or 8000 0000₁₆, as appropriate. A positive sign indicates that the phase change rate in the Parent PTP Instance is greater than that in the Slave PTP Instance. The measurement of this value is optional, but if not measured, the value of parentDS.parentStats shall be FALSE.

The data type shall be Integer32. The initialization value shall be 7FFF FFFF₁₆ irrespective of whether the computation is implemented in the PTP Instance.

A value equal to 7FFF FFFF₁₆ indicates that either the value exceeds the capacity of the data type or that the value has not been computed.

NOTE—This value is dependent on the measurement time interval used. When this value is critical for an application domain, the time interval is specified in the applicable PTP Profile.

8.2.3.6 parentDS.grandmasterIdentity

The value of parentDS.grandmasterIdentity is the value of the clockIdentity attribute (see 7.6.2.2) of the Grandmaster PTP Instance.

The data type shall be ClockIdentity. The initialization value shall be the defaultDS.clockIdentity member.

8.2.3.7 parentDS.grandmasterClockQuality

The value of parentDS.grandmasterClockQuality is the value of the clockQuality attribute (see 7.6.2.5, 7.6.2.6, and 7.6.2.7) of the Grandmaster PTP Instance.

The data type shall be ClockQuality. The initialization value shall be the value of the defaultDS.clockQuality member.

8.2.3.8 parentDS.grandmasterPriority1

The value of parentDS.grandmasterPriority1 is the value of the priority1 attribute (see 7.6.2.3) of the Grandmaster PTP Instance.

The data type shall be UInteger8. The initialization value shall be the value of the defaultDS.priority1 member.

8.2.3.9 parentDS.grandmasterPriority2

The value of parentDS.grandmasterPriority2 is the value of the priority2 attribute (see 7.6.2.4) of the Grandmaster PTP Instance.

The data type shall be UInteger8. The initialization value shall be the value of the defaultDS.priority2 member.

8.2.3.10 parentDS.protocolAddress

The value of parentDS.protocolAddress is the descriptionPortDS.protocolAddress of the PTP Port that issues the Sync messages used in synchronizing this PTP Instance. This data set member is dynamic. The data type shall be PortAddress.

The initialization value shall be as follows:

The parentDS.protocolAddress is set to the descriptionPortDS.protocolAddress with portDS.portNumber set to 0.

NOTE 1—Depending on the rules for the network transport protocol, a PTP Node with more than one PTP Port might have different protocolAddress values for each PTP Port, or they might all be the same.

NOTE 2—The value of the protocolAddress of a PTP Port on the Master PTP Instance is not present in any of the PTP event or general messages unless the PROTOCOL_ADDRESS TLV is attached. In this case, it needs to be obtained via a management mechanism or by means outside of this standard. As a result, parentDS.protocolAddress is not automatically updated when the BMCA is invoked, and its value might not be current.

8.2.3.11 parentDS.synchronizationUncertain

This data set member is optional. If the applicable PTP Profile specifies use of the synchronizationUncertain flag in Announce, this data set member shall be the value of the synchronizationUncertain flag of the most recent Announce message received from the Parent PTP Instance (see 8.2.2.5) on the PTP Port in the SLAVE or UNCALIBRATED state. If none of the PTP Ports in the PTP Instance are in SLAVE or UNCALIBRATED state, the value of synchronizationUncertain shall be FALSE.

The data type shall be Boolean. The specification initialization value shall be FALSE.

8.2.4 timePropertiesDS data set member specifications

8.2.4.1 General

The name and operational conformance, as defined in 8.1.6, of each timePropertiesDS member is summarized in Table 11.

Table 11—timePropertiesDS operational conformance

Name	OC	BC	E2E TC	P2P TC
currentUtcOffset	required	required	N/A	N/A
currentUtcOffsetValid	required	required	N/A	N/A
leap59	required	required	N/A	N/A
leap61	required	required	N/A	N/A
timeTraceable	required	required	N/A	N/A
frequencyTraceable	required	required	N/A	N/A
ptpTimescale	required	required	N/A	N/A
timeSource	required	required	N/A	N/A

All members of the timePropertiesDS data set are dynamic.

The timePropertiesDS.ptpTimescale member shall be initialized before the other members of this data set.

A Grandmaster PTP Instance updates the members of this data set (see 9.4). Other PTP Instances update the members of this dataset based upon information received in PTP messages (see 9.3.5).

NOTE—See B.4 for additional discussion on the meaning and uses of the attributes of the timePropertiesDS data set.

8.2.4.2 timePropertiesDS.currentUtcOffset

The value of timePropertiesDS.currentUtcOffset shall be the value of <dLS> received from the Grandmaster PTP Instance and shall be the value specified in Table 30 and Table 33. If the value of timePropertiesDS.currentUtcOffsetValid is FALSE, the value timePropertiesDS.currentUtcOffset shall not be utilized to compute UTC time from TAI time. The value shall be in units of seconds.

If the timescale is ARB, the value cannot be used to compute UTC from the timescale of the domain.

NOTE 1—If it is necessary to compute UTC when using the timescale ARB, the ALTERNATE_TIME_OFFSET_INDICATOR TLV of 16.3 can be used, provided that the offsets of the timescale ARB from UTC are known to the Grandmaster PTP Instance.

NOTE 2—Close to a leap-second occurrence, the value of timePropertiesDS.currentUtcOffset can differ from <dLS>; see 9.4.

The data type shall be Integer16. The initialization value shall be selected as follows:

- If the timescale is PTP, the value of timePropertiesDS.currentUtcOffset shall be the offset of TAI from UTC that is, <dLS> (see 7.2.4) when known; otherwise the value has no meaning (see 8.2.4.3).
- If the timescale is ARB, the value is implementation specific (see 7.2.4).

NOTE 3—A PTP Instance that is designed to be clockClass 6 can include implementation-specific measures to ensure it meets the specifications of 7.6.2.5 for clockClass 6 and therefore has access to the UTC offset value before initializing the timePropertiesDS.currentUtcOffset member. For example, the PTP Instance can synchronize its Local PTP Clock to the GPS system as part of the activities in the INITIALIZATION state prior to initializing the currentUtcOffset member.

8.2.4.3 timePropertiesDS.currentUtcOffsetValid

The value of timePropertiesDS.currentUtcOffsetValid shall be TRUE if the value of the timePropertiesDS.currentUtcOffset and the values of the timePropertiesDS.leap59 and timePropertiesDS.leap61 are known to be correct otherwise it shall be FALSE.

The data type shall be Boolean. The initialization value shall be TRUE if the value of timePropertiesDS.currentUtcOffset and the values of the timePropertiesDS.leap59 and timePropertiesDS.leap61 are known to be correct at the time of initialization; otherwise, it shall be FALSE.

8.2.4.4 timePropertiesDS.leap59

If the timescale is PTP, a TRUE value for timePropertiesDS.leap59 shall indicate that the last minute of the current UTC day contains 59 seconds.

If the timescale is not PTP, the value shall be set to FALSE.

The data type shall be Boolean. The initialization value shall be selected as follows:

- a) If the timescale is PTP, the value shall be the value obtained from a reference if the value from the reference is known at the time of initialization, else the value shall be FALSE.
- b) If the timescale is ARB, the value shall be FALSE.

8.2.4.5 timePropertiesDS.leap61

If the timescale is PTP, a TRUE value for timePropertiesDS.leap61 shall indicate that the last minute of the current UTC day contains 61 seconds.

If the epoch is not PTP, the value shall be set to FALSE.

The data type shall be Boolean. The initialization value shall be selected as follows:

- a) If the timescale is PTP, the value is the value obtained from a reference if the value from the reference is known at the at the time of initialization, else the value shall be FALSE.
- b) If the timescale is ARB, the value shall be FALSE.

8.2.4.6 timePropertiesDS.timeTraceable

The value of timePropertiesDS.timeTraceable shall be TRUE if the timescale is traceable to a primary reference; otherwise, the value shall be FALSE.

The data type shall be Boolean. The initialization value shall be selected as follows:

- a) The value of timePropertiesDS.timeTraceable shall be TRUE if the timescale is traceable to a primary reference at the time of initialization, else
- b) The value shall be FALSE.

The uncertainty specifications appropriate to the evaluation of whether traceability (see 3.1.80) to a primary reference is achieved should be defined in the applicable PTP Profile. In the absence of such a definition, the value of timePropertiesDS.timeTraceable is implementation specific.

8.2.4.7 timePropertiesDS.frequencyTraceable

The value of timePropertiesDS.frequencyTraceable shall be TRUE if the frequency determining the timescale is traceable to a primary reference; otherwise, the value shall be FALSE.

The data type shall be Boolean. The initialization value shall be selected as follows:

- a) If the frequency is traceable to a primary reference at the time of initialization the value shall be TRUE, else
- b) The value shall be FALSE.

The uncertainty specifications appropriate to the evaluation of whether traceability (see 3.1.80) to a primary reference is achieved should be defined in the applicable PTP Profile. In the absence of such a definition, the value of timePropertiesDS.frequencyTraceable is implementation specific.

8.2.4.8 timePropertiesDS.ptpTimescale

The value of timePropertiesDS.ptpTimescale is TRUE if the timescale (see 7.2.1) of the Grandmaster PTP Instance is PTP and FALSE otherwise.

The data type shall be Boolean. The initialization value shall be selected as follows:

- a) If the PTP Instance timescale (see 7.2.1) is PTP and this is known at the time of initialization, the value shall be set to TRUE, else
- b) The value shall be FALSE, indicating that the timescale is ARB.

8.2.4.9 timePropertiesDS.timeSource

The value of timePropertiesDS.timeSource is the source of time used by the Grandmaster PTP Instance.

The data type shall be Enumeration8. The initialization value shall be selected as follows:

- a) If the PTP Instance timeSource (see 7.6.2.8) is known at the time of initialization, the value shall be set to that value, else
- b) The value shall be INTERNAL_OSCILLATOR.

8.2.5 descriptionDS

8.2.5.1 General

The name and operational conformance, as defined in 8.1.6, of each descriptionDS member is summarized in Table 12.

Table 12—descriptionDS operational conformance

Name	OC	BC	E2E TC	P2P TC
manufacturerIdentity	management	management	management	management
productDescription	management	management	management	management
productRevision	management	management	management	management
userDescription	management	management	management	management

The descriptionDS data set provides descriptive information for the PTP Instance.

This data set may be supported for management (see 8.1.6). For the data set and each of its members, operational conformance is “management” (as defined in 8.1.6) for all types of PTP Instance.

8.2.5.2 descriptionDS.manufacturerIdentity

This member may be supported for management (see 8.1.6).

The data type shall be Octet[3]. The classification is static.

For management read, the value of manufacturerIdentity shall be an OUI or CID owned by the manufacturer of the PTP Instance.¹⁴

For example:

The OUI for Organization *X* is 001B19 (hex). The byte and bit representations of the manufacturerIdentity of Company *X* are illustrated in Table 13.

Table 13—Example—Organization X OUI

OUI or CID				
<i>addr+0</i>	<i>addr+1</i>	<i>addr+2</i>		<i>order</i>
00	1B	19		hex
00000000	00011011	00011001		bits
↑ most significant bit	↑ group address bit		↑ least significant bit	
most significant byte		least significant byte		

8.2.5.3 descriptionDS.productDescription

This member may be supported for management (see 8.1.6).

The data type shall be PTPTText. The classification is static.

For management read, the productDescription field shall indicate, in order:

- The name of the manufacturer of the PTP Instance, manufacturerName, followed by a semicolon (;)
- The model number of the PTP Instance, modelNumber, followed by a semicolon (;)
- An identifier of this PTP Instance, instanceIdentifier, such as the MAC address or the serial number

The maximum number of symbols in productDescription shall be 64.

The content and meaning of the manufacturerName, modelNumber, and the instanceIdentifier strings are determined by the manufacturer of the PTP Instance.

¹⁴ OUI and CIDs [B12] can be obtained from the IEEE Registration Authority (<https://standards.ieee.org/regauth/>).

8.2.5.4 descriptionDS.productRevision

This member may be supported for management (see 8.1.6).

The data type shall be PTPText. The classification is static.

For management read, the value shall indicate the revisions for PTP Instance's hardware (HW), firmware (FW), and software (SW). This information shall be semicolon (;) separated text fields in the order HW;FW;SW. Nonapplicable elements shall be indicated by a text fields of zero length.

The maximum number of symbols in productRevision shall be 32.

8.2.5.5 descriptionDS.userDescription

This member may be supported for management (see 8.1.6).

The data type shall be PTPText. The classification is configurable.

The userDescription field should specify, in order:

- a) A user-defined name of the PTP Instance, for example, Sensor-1, followed by a semicolon (;)
- b) A user-defined physical location of the PTP Instance, for example, Rack-2 Shelf-3

Either field may be absent, for example, (Rack-2 Shelf-3) or (Sensor-1). By default, no text is required (i.e., empty string).

The maximum number of symbols in userDescription shall be 128.

8.2.6 faultLogDS

8.2.6.1 General

The name and operational conformance, as defined in 8.1.6, of each faultLogDS member is summarized in Table 14.

Table 14—faultLogDS members

Name	OC	BC	E2E TC	P2P TC
numberOfFaultRecords	management	management	management	management
faultRecordList	management	management	management	management
Reset	management	management	management	management

This data set represents an optional mechanism for logging of faults that occur in the PTP Instance. If one member of the faultLogDS is supported, all members of the faultLogDS shall be supported.

8.2.6.2 faultLogDS.numberOfFaultRecords

This member may be supported for management (see 8.1.6).

The value of the numberOfFaultRecords member shall be the number of fault records available in faultRecordList (8.2.6.3).

The data type shall be UInteger16. The classification is dynamic.

8.2.6.3 faultLogDS.faultRecordList

This member may be supported for management (see 8.1.6).

The faultRecordList shall consist of an array of numberOfFaultRecords (see 8.2.6.2) elements, where each element shall be a structure of data type FaultRecord (see 5.3.10). The classification is dynamic.

The value of faultRecordList[].faultTime shall indicate the time the fault occurred as indicated by the Timestamping Clock of the PTP Instance. A value of all 1's for the fields in the timestamp shall indicate that the occurrence time is not available.

The value of the faultRecordList[].severityCode member shall be selected from the enumeration in Table 15.

Table 15—Fault log severityCode enumeration

Value (hex)	faultRecordList[].severityCode description
00	Emergency: system is unusable
01	Alert: immediate action needed
02	Critical: critical conditions
03	Error: error conditions
04	Warning: warning conditions
05	Notice: normal but significant condition
06	Informational: informational messages
07	Debug: debug-level messages
08 to FF	Reserved

The values of faultRecordList[].faultName, faultRecordList[].faultValue, and faultRecordList[].faultDescription members are implementation specific and may be of zero length.

The faultRecordList[].faultName shall be a name for the fault unique within the implementation.

The faultRecordList[].faultValue shall be any value that may be associated with the fault that is necessary for fault diagnosis.

The faultRecordList[].faultDescription shall be any supplementary description of the fault.

The maximum size of faultRecordList is implementation specific. The faultRecordList is maintained by the PTP Instance until a faultLogDS.reset (8.2.6.4) is written to TRUE.

8.2.6.4 faultLogDS.reset

This member may be supported for management (see 8.1.6).

This member is used to reset the fault log.

The data type shall be Boolean. The classification is configurable.

For management read, this member always returns FALSE.

A management write of the value TRUE shall cause the contents of faultRecordList (8.2.6.3) to be cleared, and numberOfFaultRecords (8.2.6.2) to be set to zero.

A management write of the value FALSE has no effect, and the PTP Instance should return a management error.

NOTE—It is possible that new faults can occur between management read of faultLogDS.faultRecordList and management write of faultLogDS.reset. Handling of this scenario is implementation specific. For example, the PTP Instance could copy the new faults into faultLogDS.faultRecordList after performing the reset, to ensure that they are not lost.

8.2.7 nonvolatileStorageDS

8.2.7.1 General

The name and operational conformance, as defined in 8.1.6, of each nonVolatileStorageDS member is summarized in Table 16.

Table 16—nonVolatileStorageDS members

Name	OC	BC	E2E TC	P2P TC
reset	management	management	management	management
save	management	management	management	management

The nonvolatileStorageDS data set provides configuration of nonvolatile storage for the PTP Instance.

For the data set and each of its members, operational conformance is “management” (as defined in 8.1.6) for all types of PTP Instance.

8.2.7.2 nonVolatileStorageDS.reset

This member may be supported for management (see 8.1.6).

The data type shall be Boolean. The classification is configurable.

For management read, this member always returns FALSE.

When management write is supported, a write of the value TRUE shall cause the contents of nonvolatile read-write storage to be reset to the applicable dynamic or configurable data set initialization values as specified in 8.1.3.5.

When management write is supported, a write of the value FALSE has no effect, and the PTP Instance should return a management error.

8.2.7.3 nonVolatileStorageDS.save

This member may be supported for management (see 8.1.6).

The data type shall be Boolean. The classification is configurable.

For management read, this member always returns FALSE.

When management write is supported, a write of the value TRUE shall cause the current values of the applicable dynamic or configurable data set members to be copied into nonvolatile read-write storage as specified in 8.1.3.5.

When management write is supported, a write of the value FALSE has no effect, and the PTP Instance should return a management error.

8.2.8 pathTraceDS

This data set provides management access to the pathTraceDS of the optional path trace mechanism (see 16.2). The pathTraceDS is specified in 16.2.2.

8.2.9 alternateTimescaleOffsetsDS

This data set provides management access to the alternateTimescaleOffsetsDS of the optional alternate timescale offsets mechanism (see 16.3). The alternateTimescaleOffsetsDS is specified in 16.3.4.

8.2.10 holdoverUpgradeDS

8.2.10.1 General

The name and operational conformance, as defined in 8.1.6, of each holdoverUpgradeDS member is summarized in Table 17.

Table 17—holdoverUpgradeDS member

Name	OC	BC	E2E TC	P2P TC
enable	optional	optional	N/A	N/A

This data set provides management access to the optional holdover upgrade mechanism (see 16.4).

8.2.11 grandmasterClusterDS

This data set provides management access to the grandmasterClusterDS of the optional grandmaster cluster mechanism (see 17.2). The grandmasterClusterDS is specified in 17.2.3.

8.2.12 acceptableMasterTableDS

This data set provides management access to the acceptableMasterTableDS of the optional acceptable master table mechanism (see 17.5). The acceptableMasterTableDS is specified in 17.5.3.

8.2.13 performanceMonitoringDS

This data set provides management access to the performanceMonitoringDS of the optional performance monitoring feature (see Annex J). The performanceMonitoringDS is specified in J.5.1.

8.2.14 enhancedSynchronizationAccuracyMetricsDS

8.2.14.1 General

The name and operational conformance (see 8.1.6) of each enhancedSynchronizationAccuracyMetricsDS member is summarized in Table 18.

Table 18—enhancedSynchronizationAccuracyMetricsDS members

Name	OC	BC	E2E TC	P2P TC
enable	optional	optional	optional	optional

This data set provides management access to the Enhanced Synchronization Accuracy Metrics (see 16.12) option.

8.2.14.2 enhancedSynchronizationAccuracyMetricsDS.enable

The data type shall be Boolean. The classification is configurable. The specification initialization value (see 8.1.3.4) shall be FALSE.

If the Enhanced Synchronization Accuracy Metrics, 16.12, is implemented, the value TRUE shall indicate that the option is enabled on the PTP Instance, and the value FALSE shall indicate that the option is disabled on the PTP Instance.

8.2.15 portDS data set member specifications

8.2.15.1 General

Each portDS data set corresponds to a single PTP Port of the PTP Instance.

The name and operational conformance, as defined in 8.1.6, of each portDS member is summarized in Table 19.

Table 19—portDS operational conformance

Name	OC	BC	E2E TC	P2P TC
portIdentity	required	required	optional ¹⁵	required
portState	required	required	N/A	N/A
logMinDelayReqInterval ¹⁶	required (optional)	required (optional)	required	N/A
meanLinkDelay ¹⁷	required	required	N/A	required
logAnnounceInterval	required	required	N/A	N/A
announceReceiptTimeout	required	required	N/A	N/A
logSyncInterval	required	required	N/A	N/A
delayMechanism ¹⁷	required (optional)	required (optional)	N/A	N/A
logMinPdelayReqInterval ¹⁷	required (optional)	required (optional)	N/A	required
versionNumber	required	required	N/A	required
minorVersionNumber	required	required	N/A	required
portEnable	optional	optional	N/A	N/A
masterOnly	optional	optional	N/A	N/A
delayAsymmetry	required	required	required	required

¹⁵ If an E2E TC supports an optional feature that initiates transmission of a PTP message (e.g., PTP management messages), data set members that are used to form sourcePortIdentity are required (e.g., portDS.portIdentity).

¹⁶ This member is required only for support of the end-to-end delay mechanism. For a PTP Instance that supports only the peer-to-peer delay mechanism, this member is optional.

¹⁷ This member is required only for support of the peer-to-peer delay mechanism. For a PTP Instance that supports only the end-to-end delay mechanism, this member is optional.

8.2.15.2 Static members of the portDS data set

8.2.15.2.1 portDS.portIdentity

The value of portDS.portIdentity shall be the value of the portIdentity attribute of the local PTP Port (see 7.5.2). The data type shall be PortIdentity.

The clockIdentity portion of a portIdentity is equal to the clockIdentity of the PTP Instance that the PTP Port is included in, and it is therefore equal among all PTP Ports inside the PTP Instance (see 7.5.2).

8.2.15.3 Dynamic members of the portDS data set

8.2.15.3.1 portDS.portState

The value of portDS.portState shall be the value of the current state of the protocol engine associated with this PTP Port (see 9.2) and shall be taken from the enumeration in Table 20. The data type shall be Enumeration8.

Table 20—PTP state enumeration

PTP state enumeration	Value (hex)
INITIALIZING	01
FAULTY	02
DISABLED	03
LISTENING	04
PRE MASTER	05
MASTER	06
PASSIVE	07
UNCALIBRATED	08
SLAVE	09
—	All other values reserved

The initialization value shall be INITIALIZING.

8.2.15.3.2 portDS.logMinDelayReqInterval

The value of portDS.logMinDelayReqInterval is the logarithm to the base 2 of the minDelayReqInterval (see 7.7.2.4). The initialization value is implementation specific, consistent with 7.7.2.4. The data type shall be Integer8.

NOTE—This value applies only to multicast messages or a unicast message where unicast negotiation has not been used. See 13.3.2.14 and Table 42 for the value placed in the logMessageInterval field of transmitted messages.

8.2.15.3.3 portDS.meanLinkDelay

If the value of the portDS.delayMechanism member is P2P (peer-to-peer), the value of portDS.meanLinkDelay shall be an estimate of the current one-way propagation delay on the PTP Link, that is, <meanLinkDelay>, attached to this PTP Port computed using the peer-to-peer delay mechanism (see 11.4). If the value of portDS.delayMechanism is COMMON_P2P, the value of portDS.meanLinkDelay shall be equal to the value of cmldsLinkPortDS.commonMeanLinkDelay Information.meanLinkDelay (see 16.6). If the value of the portDS.delayMechanism member is E2E (end-to-end), NO_MECHANISM, or SPECIAL, this member's value shall be zero. The initialization value shall be zero. The data type shall be TimeInterval.

8.2.15.4 Configurable members of the portDS data set

8.2.15.4.1 portDS.logAnnounceInterval

The value of portDS.logAnnounceInterval shall be the logarithm to the base 2 of the mean announceInterval (see 7.7.2.2). The data type shall be Integer8.

NOTE—This value applies only to multicast PTP messages. See 13.3.2.14 and Table 42 for the value placed in the logMessageInterval field of transmitted messages.

8.2.15.4.2 portDS.announceReceiptTimeout

The value of portDS.announceReceiptTimeout shall be an integral multiple of announceInterval (see 7.7.3.1). The data type shall be UInteger8.

NOTE—The announceInterval is equal to the value of $2^{\text{portDS.logAnnounceInterval}}$.

8.2.15.4.3 portDS.logSyncInterval

The value of portDS.logSyncInterval shall be the logarithm to the base 2 of the mean SyncInterval for multicast messages (see 7.7.2.3). The data type shall be Integer8.

NOTE 1—This value applies only to multicast PTP messages. See 13.3.2.14 and Table 42 for the value placed in the logMessageInterval field of transmitted messages.

NOTE 2—The rates for unicast transmissions are negotiated separately on a per PTP Port basis and are not constrained by this subclause.

8.2.15.4.4 portDS.delayMechanism

The value of portDS.delayMechanism shall indicate the path delay measuring mechanism used by the PTP Port in computing <meanDelay>. The value shall be taken from the enumeration in Table 21. The data type shall be Enumeration8. The specification initialization value (see 8.1.3.4) is implementation specific.

Table 21—Delay mechanism enumeration

Delay mechanism	Value (hex)	Specification
E2E	01	The PTP Port is configured to use the delay request-response mechanism.
P2P	02	The PTP Port is configured to use the peer-to-peer delay mechanism.
NO_MECHANISM	FE	The PTP Port does not implement the delay mechanism. This value shall not be used except when the applicable PTP Profile specifies that either: — The PTP Instance only supports frequency transfer (syntonization) and that neither path delay mechanism is to be used or — The PTP Instance participates in time transfer, but the system accuracy requirements are such that, for a segment of the system path, delays can be neglected allowing PTP Instances in that portion of the PTP Network to use the NO_MECHANISM attribute.
COMMON_P2P	03	The PTP Port is configured to use the Common Mean Link Delay Service option (see 16.6).
SPECIAL	04	Special Ports do not use either delay mechanism.

8.2.15.4.5 portDS.logMinPdelayReqInterval

The value of portDS.logMinPdelayReqInterval shall be the logarithm to the base 2 of the minPdelayReqInterval (see 7.7.2.5). The data type shall be Integer8.

8.2.15.4.6 portDS.versionNumber

The value of portDS.versionNumber shall indicate the PTP version in use on the PTP Port. The data type shall be UInteger4.

8.2.15.4.7 portDS.minorVersionNumber

The value of portDS.minorVersionNumber shall indicate the PTP minor version in use on the PTP Port. The data type shall be UInteger4.

8.2.15.4.8 portDS.delayAsymmetry

The value of portDS.delayAsymmetry shall be the value of <delayAsymmetry> applicable to the PTP Port as specified in 7.4.2. The data type shall be TimeInterval. The specification initialization value (see 8.1.3.4) shall be zero.

If the option of 16.8 is implemented and enabled, portDS.delayAsymmetry shall be dynamic. If the option of 16.8 is not implemented, or is implemented and not enabled, portDS.delayAsymmetry may be configurable. Otherwise, if the option of 16.8 is not implemented and portDS.delayAsymmetry is not configurable, it shall be static. If portDS.delayAsymmetry is configurable, the value of <delayAsymmetry> is the value of portDS.delayAsymmetry, as specified in item b) of 7.4.2.

8.2.15.5 Optional members of the portDS data set

For the following optional members, the classification (i.e., static, dynamic, or configuration) is provided in each member's specification.

8.2.15.5.1 portDS.portEnable

This member may be supported for management (see 8.1.6).

The data type shall be Boolean. The classification is configurable.

Unless stated otherwise by product specifications, the specification initialization value (see 8.1.3.4) shall be TRUE.

For management read, this member indicates if the PTP Port is enabled for PTP operation.

When management write is supported:

- Write of the value TRUE causes the DESIGNATED_ENABLED event (see 9.2.6.4) to occur even if the value was previously TRUE.
- Write of the value FALSE causes the DESIGNATED_DISABLED event (see 9.2.6.5) to occur even if the value was previously FALSE.

8.2.15.5.2 portDS.masterOnly

If the value of portDS.masterOnly is TRUE, the PTP Port shall be in the masterOnly mode (see 9.2.2.2). If the value is FALSE, the PTP Port shall not be in the masterOnly mode. The data type shall be Boolean. The classification is configurable.

The specification initialization value (see 8.1.3.4) shall be FALSE unless the PTP Port is specified-by-design to be masterOnly; in which case, the value shall be TRUE.

8.2.16 timestampCorrectionPortDS

8.2.16.1 General

The name and operational conformance, as defined in 8.1.6, of each timestampCorrectionPortDS member is summarized in Table 22.

Table 22—timestampCorrectionPortDS members

Name	OC	BC	E2E TC	P2P TC
egressLatency	Optional	optional	optional	optional
ingressLatency	Optional	optional	optional	optional

This optional data set provides access to the configurable correction of timestamps provided to the PTP Protocol (see 16.7). The timestampCorrectionPortDS applies to the PTP Port.

8.2.16.2 timestampCorrectionPortDS.egressLatency

The timestampCorrectionPortDS.egressLatency shall be the interval between the <egressProvidedTimestamp> provided for a PTP message and the time at which the message timestamp point of the PTP message crosses the reference plane. The value of timestampCorrectionPortDS.egressLatency is the value of <egressLatency>, which is used to update the provided timestamp per 7.3.4.2. The data type of timestampCorrectionPortDS.egressLatency shall be TimeInterval. The classification is configurable.

NOTE 1—The correction provided in timestampCorrectionPortDS.egressLatency is residual in the sense that the captured timestamps can be also corrected by an implementation-specific method before being provided to the PTP stack (see 7.3.4.2).

NOTE 2—The value of timestampCorrectionPortDS.egressLatency can depend on the configuration of the medium and of the physical port, for example, operating mode, speed, and the type of SFP. For high accuracy, it is important to update this data set member with the value relevant to the current configuration. Such an update is outside the scope of this standard.

NOTE 3—The data type of timestampCorrectionPortDS.egressLatency is TimeInterval (see 5.3.2); thus, it is divided by 2^{+16} to obtain the value of <egressLatency> in nanoseconds.

8.2.16.3 timestampCorrectionPortDS.ingressLatency

The timestampCorrectionPortDS.ingressLatency shall be the interval between the time the message timestamp point of an ingress PTP message crosses the reference plane and the <ingressProvidedTimestamp> provided for the PTP message. The value of timestampCorrectionPortDS.ingressLatency is the value of <ingressLatency>, which is used to update the

provided timestamp per 7.3.4.2. The data type of timestampCorrectionPortDS.ingressLatency shall be TimeInterval. The classification is configurable.

NOTE 1—The correction provided in timestampCorrectionPortDS.ingressLatency is residual in the sense that the captured timestamps can be also corrected by an implementation-specific method before being provided to the PTP stack (see 7.3.4.2).

NOTE 2—The value of timestampCorrectionPortDS.ingressLatency can depend on the configuration of the medium and of the physical port, for example, operating mode, speed, and the type of SFP. For high accuracy, it is important to update this data set member with the value relevant to the current configuration. Such an update is outside the scope of this standard.

NOTE 3—The data type of timestampCorrectionPortDS.ingressLatency is TimeInterval (see 5.3.2); thus, it is divided by 2^{+16} to obtain the value of <ingressLatency> in nanoseconds.

8.2.17 asymmetryCorrectionPortDS

8.2.17.1 General

The name and operational conformance, as defined in 8.1.6, of each asymmetryCorrectionPortDS member is summarized in Table 23.

Table 23—asymmetryCorrectionPortDS members

Name	OC	BC	E2E TC	P2P TC
constantAsymmetry	optional	optional	optional	optional
scaledDelayCoefficient	optional	optional	optional	optional
enable	optional	optional	optional	optional

This optional data set provides access to asymmetry correction parameters that are used to compute the value of the <delayAsymmetry> when the option of 16.8 is implemented and enabled. The asymmetryCorrectionPortDS applies to the PTP Port.

8.2.17.2 asymmetryCorrectionPortDS.constantAsymmetry

The value of asymmetryCorrectionPortDS.constantAsymmetry shall be the constant asymmetry used in 16.8 to fine-adjust the dynamically calculated value of <delayAsymmetry>. The data type of asymmetryCorrectionPortDS.constantAsymmetry shall be TimeInterval. The specification initialization value (see 8.1.3.4) shall be 0. The classification is configurable.

NOTE—The data type of asymmetryCorrectionPortDS.constantAsymmetry is TimeInterval (see 5.3.2); thus, it is divided by 2^{+16} to obtain its value in nanoseconds.

8.2.17.3 asymmetryCorrectionPortDS.scaledDelayCoefficient

The asymmetryCorrectionPortDS.scaledDelayCoefficient shall be the <delayCoefficient> (α) in 7.4.3 and 16.8. The data type of asymmetryCorrectionPortDS.scaledDelayCoefficient shall be RelativeDifference (see 5.3.11) and its allowed range shall be the closed range $[-2^{63} + 1; +2^{63} - 1]$. The classification is configurable.

NOTE—The data type of asymmetryCorrectionPortDS.scaledDelayCoefficient is RelativeDifference (see 5.3.11); thus, it is divided by 2^{+62} to obtain the fractional value of <delayCoefficient> (α). The specified range allows for storing values of <delayCoefficient> from $(-2^{+1} + 2^{-62})$ to $(2^{+1} - 2^{-62})$, inclusive. The intentional exclusion of -2 from the allowed range of <delayCoefficient> values avoids division by zero when this value is used in the optional feature of 16.8.

8.2.17.4 asymmetryCorrectionPortDS.enable

The data type shall be Boolean. The classification is configurable.

When supported, the value TRUE shall indicate that the mechanism of 16.8 (i.e., the calculation of the <delayAsymmetry> for certain media) is enabled on the PTP Port, and the value FALSE shall indicate that the mechanism is disabled on the PTP Port.

8.2.18 descriptionPortDS

8.2.18.1 General

The name and operational conformance, as defined in 8.1.6, of each descriptionPortDS member is summarized in Table 24.

Table 24—descriptionPortDS members

Name	OC	BC	E2E TC	P2P TC
profileIdentifier	management	management	management	management
protocolAddress	optional	optional	management	management

The descriptionPortDS data set provides descriptive information for the PTP Port.

8.2.18.2 descriptionPortDS.profileIdentifier

This member may be supported for management (see 8.1.6).

The data type shall be Octet[6]. The classification is static.

When profileIdentifier is supported, its value shall identify the PTP Profile implemented by the PTP Port, and its value shall be assigned by the creator of the PTP Profile as defined in 20.3.3.

For example:

Subclause I.4.1 defines the profileIdentifier of the “Default PTP Profile for use with the peer-to-peer delay mechanism” as the 6-octet field 001B19020100₁₆. The bit and byte representation of this profileIdentifier are shown in Table 25.

Table 25—Example—peer-to-peer default PTP Profile profileIdentifier

OUI or CID			profileIdentifier			field
<i>addr+0</i> 00	<i>addr+1</i> 1B	<i>addr+2</i> 19	<i>addr+3</i> 02	<i>addr+4</i> 01	<i>addr+5</i> 00	<i>order</i> <i>hex</i>
00000000	00011011	00011001	00000010	00000001	00000000	<i>bits</i>
↑ most signifi cant bit ↑ group addres s bit					↑ least signifi cant bit	
most significant byte					least significant byte	

8.2.18.3 descriptionPortDS.protocolAddress

The value of descriptionPortDS.protocolAddress shall be the protocol address that is used as the source address by the network transport protocol for this PTP Port.

NOTE—Depending on the rules for the network transport protocol, a PTP Node with more than one PTP Port might have different protocolAddress values for each PTP Port, or they might all be the same.

This data set member is static. The data type shall be PortAddress.

8.2.19 unicastNegotiationPortDS

8.2.19.1 General

The name and operational conformance, as defined in 8.1.6, of each unicastNegotiationDS member is summarized in Table 26.

Table 26—unicastNegotiationDS members

Name	OC	BC	E2E TC	P2P TC
enable	optional	optional	N/A	N/A

This data set provides management access to the optional unicast negotiation mechanism (16.1).

8.2.19.2 unicastNegotiationPortDS.enable

The data type shall be Boolean. The classification is configurable.

If the unicast message negotiation option (16.1) is implemented, the value TRUE shall indicate that the unicast negotiation mechanism is enabled on the PTP Port, and the value FALSE shall indicate that the unicast negotiation mechanism is disabled on the PTP Port.

8.2.20 alternateMasterPortDS

This data set provides management access to the alternateMasterPortDS of the optional alternate master mechanism (see 17.3). The alternateMasterPortDS is specified in 17.3.3.

8.2.21 unicastDiscoveryPortDS

This data set provides management access to the unicastDiscoveryPortDS of the optional unicast discovery mechanism (see 17.4). The unicastDiscoveryPortDS is specified in 17.4.3.

8.2.22 acceptableMasterPortDS

This data set provides management access to the acceptableMasterPortDS of the optional acceptable master mechanism (see 17.5). The acceptableMasterPortDS is specified in 17.5.4.

8.2.23 L1SyncBasicPortDS

This optional data set stores configurable and dynamic members that are obligatory for operation of the Layer-1 based synchronization performance enhancement optional feature defined in Annex L.

The L1SyncBasicPortDS is specified in L.5 and includes the following members:

- a) Configurable members:
 - 1) L1SyncBasicPortDS.L1SyncEnabled defined in L.5.2.1
 - 2) L1SyncBasicPortDS.txCoherentIsRequired defined in L.5.2.2
 - 3) L1SyncBasicPortDS.rxCoherentIsRequired defined in L.5.2.3
 - 4) L1SyncBasicPortDS.congruentIsRequired defined in L.5.2.4
 - 5) L1SyncBasicPortDS.optParamsEnabled defined in L.5.2.5
 - 6) L1SyncBasicPortDS.logL1SyncInterval defined in L.5.2.6
 - 7) L1SyncBasicPortDS.L1SyncReceiptTimeout defined in L.5.2.7
- b) Dynamic members:
 - 1) L1SyncBasicPortDS.L1SyncLinkAlive defined in L.5.3.1
 - 3) L1SyncBasicPortDS.isTxCoherent defined in L.5.3.2
 - 4) L1SyncBasicPortDS.isRxCoherent defined in L.5.3.3
 - 5) L1SyncBasicPortDS.isCongruent defined in L.5.3.4
 - 6) L1SyncBasicPortDS.L1SyncState defined in L.5.3.5
 - 7) L1SyncBasicPortDS.peerTxCoherentIsRequired defined in L.5.3.6
 - 8) L1SyncBasicPortDS.peerRxCoherentIsRequired defined in L.5.3.7
 - 9) L1SyncBasicPortDS.peerCongruentIsRequired defined in L.5.3.8
 - 10) L1SyncBasicPortDS.peerIsTxCoherent defined in L.5.3.9
 - 11) L1SyncBasicPortDS.peerIsRxCoherent defined in L.5.3.10
 - 12) L1SyncBasicPortDS.peerIsCongruent defined in L.5.3.11

8.2.24 L1SyncOptParamsPortDS

This optional data set stores configurable and dynamic members that are optional for operation of the Layer-1 based synchronization performance enhancement optional feature defined in Annex L.

The L1SyncOptParamsPortDS is specified in L.8.4 and includes the following members:

- a) Configurable members:
 - 1) L1SyncOptParamsPortDS.timestampsCorrectedTx defined in L.8.4.2.1
- b) Dynamic members:
 - 1) L1SyncOptParamsPortDS.phaseOffsetTxValid defined in L.8.4.3.1
 - 13) L1SyncOptParamsPortDS.frequencyOffsetTxValid defined in L.8.4.3.2
 - 14) L1SyncOptParamsPortDS.phaseOffsetTx defined in L.8.4.3.3
 - 15) L1SyncOptParamsPortDS.phaseOffsetTxTimestamp defined in L.8.4.3.4
 - 16) L1SyncOptParamsPortDS.frequencyOffsetTx defined in L.8.4.3.5
 - 17) L1SyncOptParamsPortDS.frequencyOffsetTxTimestamp defined in L.8.4.3.6

8.2.25 communicationCapabilitiesPortDS

The dataset members of communicationCapabilitiesPortDS are static.

All members of this data set shall have data type CommunicationCapabilities.

For all members of this data set:

- The value of communicationCapabilities.multicastCapable shall be TRUE if the PTP Port is capable of transmitting PTP messages using multicast communication; otherwise it shall be FALSE.
- The value of communicationCapabilities.unicastCapable shall be TRUE if the PTP Port is capable of transmitting PTP messages using unicast communication; otherwise it shall be FALSE.
- The value of communicationCapabilities.unicastNegotiationCapable shall be TRUE if the PTP Port is capable of negotiating unicast communication using the unicast negotiation option (see 16.1), and the unicastNegotiationPortDS.enable (see 8.2.19.2) is TRUE; otherwise the value of communicationCapabilities.unicastNegotiationCapable shall be FALSE.
- The value of communicationCapabilities.unicastNegotiationRequired shall be TRUE if the value of communicationCapabilities.unicastNegotiationCapable is TRUE and the use of the unicast negotiation option of 16.1 is required by the implementation; otherwise it shall be FALSE.

8.2.25.1 communicationCapabilitiesPortDS.syncCapabilities

The value of communicationCapabilitiesPortDS.syncCapabilities shall be the CommunicationCapabilities attributes of the local PTP Port with respect to sending Sync messages.

8.2.25.2 communicationCapabilitiesPortDS.delayRespCapabilities

The value of communicationCapabilitiesPortDS.delayRespCapabilities shall be the CommunicationCapabilities attributes of the local PTP Port with respect to sending Delay_Resp messages.

8.2.26 performanceMonitoringPortDS

This data set provides management access to the performanceMonitoringPortDS of the optional performance monitoring feature (see Annex J). The performanceMonitoringPortDS is specified in J.5.2.

8.2.27 commonServicesPortDS

This data set provides management access to the common services ports that are available to a PTP Port, and it is specified in 16.6.5.

8.2.28 externalPortConfigurationPortDS

This data set provides management access to the external configuration option for each PTP Port, and it is specified in 17.6.3.

8.2.29 slaveMonitoringPortDs

This data sets provides management access to the optional Slave Event Monitor service, and it is specified in 16.11.6.

8.3 Data sets for Transparent Clocks

8.3.1 General

According to the principles specified in 7.1, each data set is scoped to a single domain and, therefore, to a single PTP Instance. In the 2008 edition, although it was not clearly specified, the data sets for Transparent Clocks were typically interpreted as applicable to the PTP Node. When a PTP Node contains a single PTP Instance, this interpretation aligned with the design principles of 7.1. When a PTP Node contains more than one PTP Instance, this interpretation has limitations that are not acceptable as the standard progresses.

8.3.1.1 Node-level data sets for Transparent Clocks

The 2008 edition of this standard specified the following data sets for each Transparent Clock:

- transparentClockDefaultDS data set (see 8.3.2)
- transparentClockPortDS data set (one data set for each PTP Port; see 8.3.3)

This edition of the standard specifies transparentClockDefaultDS and transparentClockPortDS as applying to the PTP Node, but it designates those data sets as deprecated (see 4.2.8).

These data sets operate on all domains, and therefore, they represent multiple PTP Instances (i.e., multiple Transparent Clocks).

8.3.1.2 Instance-level data sets for Transparent Clocks

This edition of the standard recommends that Transparent Clocks use the data sets for PTP Instance (see 8.2). The PTP Node can provide one or more Transparent Clocks (each with an entry in instanceList), and each Transparent Clock supports a single PTP Instance and domain.

Each instance-level Transparent Clock contains all the mandatory data sets of 8.2, and all the optional data sets of those options that it supports. Each data set in each instance-level Transparent Clock contains those members that are applicable to the Transparent Clock, depending on whether it is end-to-end or peer-to-peer, in the respective operational conformance table.

8.3.2 transparentClockDefaultDS data set member specifications (deprecated)

8.3.2.1 General

The members of this data set are as follows:

- transparentClockDefaultDS.clockIdentity
- transparentClockDefaultDS.numberPorts
- transparentClockDefaultDS.delayMechanism
- transparentClockDefaultDS.primaryDomain

8.3.2.2 Static members of the transparentClockDefaultDS data set

8.3.2.2.1 transparentClockDefaultDS.clockIdentity

The value of transparentClockDefaultDS.clockIdentity shall be the value of the clockIdentity attribute (see 7.6.2.2) of the local clock (see NOTE below). The data type shall be ClockIdentity.

NOTE—The “local clock” phrase was transferred from IEEE Std 1588-2008, and it is not used in this revision. In the context of this revision, “local clock” can be interpreted as belonging to the PTP Node (i.e., all PTP Instances) or a single PTP Instance.

When any member of transparentClockDefaultDS is supported for management, the transparentClockDefaultDS.clockIdentity member shall be supported for management. If a management read of transparentClockDefaultDS.clockIdentity succeeds, the managing entity can assume that Transparent Clock management applies to the entire PTP Node (see 8.3.1.1). If a management read of transparentClockDefaultDS.clockIdentity fails, the managing entity can assume that if Transparent Clock(s) are supported, Transparent Clock management applies to distinct PTP Instances (see 8.3.1.2).

8.3.2.2.2 transparentClockDefaultDS.numberPorts

The value of transparentClockDefaultDS.numberPorts shall be the number of PTP Ports. The data type shall be UInteger16.

8.3.2.3 Configurable members of the transparentClockDefaultDS data set

8.3.2.3.1 transparentClockDefaultDS.delayMechanism

If the Transparent Clock is an end-to-end Transparent Clock, the value of the transparentClockDefaultDS.delayMechanism shall be E2E; see Table 21. If the Transparent Clock is a peer-to-peer Transparent Clock, the value shall be P2P. The data type shall be Enumeration8.

8.3.2.3.2 transparentClockDefaultDS.primaryDomain

The value of transparentClockDefaultDS.primaryDomain shall be the value of domainNumber of the primary syntonization domain. The specification initialization value (see 8.1.3.4) shall be 0. The data type shall be UInteger8.

NOTE—In IEEE Std 1588-2008, the term “primary syntonization domain” is defined to be domainNumber 0, or if the default is configured to a new domain, the configured default domain.

8.3.3 transparentClockPortDS data set member specifications (deprecated)

8.3.3.1 General

The members of this data set are as follows:

- transparentClockPortDS.portIdentity
- transparentClockPortDS.logMinPdelayReqInterval
- transparentClockPortDS.faultyFlag
- transparentClockPortDS.peerMeanPathDelay

8.3.3.2 Static members of the portDS data set

8.3.3.2.1 transparentClockPortDS.portIdentity

The value of transparentClockPortDS.portIdentity shall be the value of the portIdentity attribute of the local port (see NOTE below and 7.5.2). The data type shall be PortIdentity.

NOTE—The “local port” phrase was transferred from IEEE Std 1588-2008, and it is not used in this revision. In the context of this revision, “local port” can be interpreted as belonging to the PTP Node (i.e., all PTP Instances) or to a single PTP Instance.

8.3.3.3 Dynamic members of the portDS data set

8.3.3.3.1 transparentClockPortDS.logMinPdelayReqInterval

The value of transparentClockPortDS.logMinPdelayReqInterval shall be the logarithm to the base 2 of the minPdelayReqInterval (see 7.7.2.5). The data type shall be Integer8.

8.3.3.3.2 transparentClockPortDS.faultyFlag

The value of transparentClockPortDS.faultyFlag shall be TRUE if the PTP Port is faulty and FALSE if the PTP Port is operating normally. The initialization value shall be FALSE. The data type shall be Boolean.

8.3.3.3.3 transparentClockPortDS.peerMeanPathDelay

If the value of the transparentClockDefaultDS.delayMechanism member is P2P, the value of transparentClockPortDS.peerMeanPathDelay shall be the estimate of the current one-way propagation delay, that is, <meanLinkDelay> on the PTP Link attached to this PTP Port computed using the peer-to-peer delay mechanism (see 11.4).

NOTE—The transparentClockPortDS.peerMeanPathDelay is specified as applying to the PTP Node; thus, it is calculated for a PTP Node according to the specification of portDS.meanLinkDelay.

If the value of the transparentClockDefaultDS.delayMechanism member is E2E, the value shall be zero. The data type is TimeInterval. The initialization value shall be zero.

8.4 commonMeanLinkDelayService data sets

These data sets provide management access to the optional Common Mean Link Delay Service, and they are specified in 16.6.4.

9. PTP for Ordinary Clocks and Boundary Clocks

9.1 General protocol requirements for PTP Ordinary Clocks and Boundary Clocks

Ordinary Clocks and Boundary Clocks:

- a) They may operate within more than one domain (see 7.1). The operation of each domain shall be independent of the others.
- b) When required by the state machine of 9.2, they shall synchronize per 12.3.
- c) They shall perform one of the following:
 - 1) Correct for path delay using one of the following options:
 - i) Delay request-response mechanism (see 11.3).
 - ii) Peer-to-peer delay mechanism (see 11.4).
 - iii) The Common Mean Link Delay Service option (see 16.6) or
 - 2) Under the conditions specified for the NO_MECHANISM option of Table 21 in 8.2.15.4.4, not use any of the above path delay mechanisms.

A PTP Port may implement both the delay request-response and the peer-to-peer delay mechanisms provided only one mechanism is active at the same time.

PTP Instances synchronize only to the PTP Instance selected using the best master clock algorithm.

An Ordinary Clock or Boundary Clock that receives an Announce, Sync, Follow_Up, or Delay_Resp message in which the value of the header flag, alternateMasterFlag, is TRUE shall discard the PTP message except as provided in 17.3.

An Ordinary Clock shall contain a single PTP Port, obeying the requirements of 9.2.

A Boundary Clock shall contain multiple PTP Ports, each obeying the requirements of 9.2.

When using the cumulative frequency transfer option defined in 16.10, the specifications of 9.5.4, 9.5.5, 9.5.9, and 9.5.10 are modified for Boundary Clocks and shall follow the modifications specified in 16.10.4.4.

9.2 State protocol

9.2.1 General state requirements

All Ordinary Clocks and Boundary Clocks shall implement the state machine and state behavior of 9.2.

9.2.2 PTP Instances with special state behavior

9.2.2.1 Slave-Only Ordinary Clocks

An Ordinary Clock may be designed to be a slaveOnly PTP Instance. An implementation may optionally provide the ability to configure an Ordinary Clock to a slaveOnly mode via the member defaultDS.slaveOnly of 8.2.1.4.4 or by implementation-specific means. A slaveOnly PTP Instance shall implement the state machine illustrated in Figure 31.

NOTE 1— A slaveOnly PTP Instance can never enter the MASTER state. It is recommended therefore that PTP Networks contain at least one non-slaveOnly PTP Instance. A slaveOnly PTP Instance uses a different state machine than a non-slaveOnly PTP Instance and has a different clockClass number (see 7.6.2.5).

The PTP Port of a slaveOnly Ordinary Clock should not be in the masterOnly mode.

NOTE 2—A slaveOnly Ordinary Clock with its port in masterOnly mode will not process incoming Announce messages, and it will therefore maintain a degenerate operational condition not generally of use within the PTP network. Some management implementations might be simplified by allowing slaveOnly mode and masterOnly mode to be temporarily active at the same time.

9.2.2.2 MasterOnly PTP Ports

An implementation may optionally provide the ability to configure the PTP Port on an Ordinary Clock or any PTP Port on a Boundary Clock to a masterOnly mode via the member portDS.masterOnly of 8.2.15.5.2 or by implementation-specific means. The PTP Port of an Ordinary Clock or a Boundary Clock may be designed to be a masterOnly PTP Port.

Announce messages received on a masterOnly PTP Port shall not be considered in the operation of the best master clock algorithm or in the update of data sets.

NOTE 1—The result of this requirement is that E_{rbest} on a masterOnly PTP Port will be the empty set and therefore will never be E_{best} . From Figure 30 these conditions ensure that the PTP Port will never be in the SLAVE or PASSIVE states and any disregarded Announce messages will never influence the update of data sets. The use of the masterOnly feature can result in potentially undesirable situations.

NOTE 2—A masterOnly PTP Port can never enter the SLAVE state.

The PTP Port of a slaveOnly Ordinary Clock should not be in the masterOnly mode.

NOTE 3—A slaveOnly Ordinary Clock utilizes the slaveOnly state machine, which does not enable transition to the MASTER state. If its port is in masterOnly mode, and therefore not processing incoming Announce messages, it will

maintain a degenerate operational condition not generally of use within the PTP network. Some management implementations might be simplified by allowing slaveOnly mode and masterOnly mode to be temporarily active at the same time.

9.2.2.3 PTP Ports not on a Slave-Only Ordinary Clock or in a masterOnly mode

PTP Ports not on a Slave-Only Ordinary Clock or in a masterOnly mode shall implement the state machine illustrated in Figure 30.

9.2.3 Non-slaveOnly PTP Instances

A Boundary Clock shall not be a slaveOnly PTP Instance. Ordinary Clocks not designed or configured as slaveOnly and Boundary Clocks shall implement the state machine illustrated in Figure 30.

9.2.4 State definitions

The behavior of the states of a PTP Port shall be as defined in Table 27 with the exception of the provisions for unicast PTP messages specified in 16.1.

When the specifications of 17.6 are disabled or not implemented, the behavior of the states of a PTP Port is associated with the state machines of Figure 30 and Figure 31.

When the specifications of 17.6 are enabled, the state machines of Figure 30 and Figure 31 are not used (see 17.6).

The actual value of the port state of a PTP Port shall be the value of portDS.portState.

NOTE 1—The restrictions on the types of PTP messages that can be placed on the communication path are not applicable for unicast PTP messages according to the provisions for unicast PTP messages specified in 16.1. Unicast negotiation and service transmission is permitted provided the operation of the protocol is preserved (see 7.3.1).

NOTE 2—Subclause 17.3 permits the use of PTP messages beyond the restrictions of Table 27.

With the exception of the DISABLED state, a PTP Port may make use of any information received in PTP messages provided such use does not violate the requirements of the protocol.

9.2.5 State machines

The state machines of this subclause shall determine the allowed transitions on the PTP Ports of Boundary Clocks and the PTP Port of an Ordinary Clock.

Subclause 4.3 specifies the notation used in Figure 30 and Figure 31.

Table 27—PTP portState definition

PTP portState	Description
INITIALIZING	A PTP Port in the INITIALIZING state as a result of a POWERUP or INITIALIZE or DESIGNATED_ENABLED or a FAULT_CLEARED event shall initialize its PTP Port data sets, hardware, and communication facilities. The PTP Port shall not place any PTP messages on its communication path. A POWERUP or INITIALIZE event results in all PTP Ports on a PTP Instance transitioning to the INITIALIZING state (see Figure 30). In this case in addition to the actions taken by the PTP Port, the PTP Instance shall initialize its PTP Instance data sets, hardware, and communication facilities.
FAULTY	The fault state of the protocol. Except for management messages that are a required response to a message received from the applicable management mechanism, a PTP Port in this state shall not transmit any PTP related messages. In a Boundary Clock, no activity on a faulty PTP Port shall affect the other PTP Ports of the PTP Instance. If fault activity on a PTP Port in this state cannot be confined to the faulty PTP Port, then all PTP Ports shall be in the FAULTY state.
DISABLED	Except for management messages that are a required response to a message received from the applicable management mechanism, a PTP Port in this state shall not transmit any PTP related messages. In a Boundary Clock, no activity at the PTP Port shall be allowed to affect the activity at any other PTP Port of the Boundary Clock. A PTP Port in this state shall discard all received PTP messages except for PTP management messages.
LISTENING	The PTP Port is waiting for the announceReceiptTimeout to expire or to receive an Announce message from a Master PTP Instance. The purpose of this state is to allow orderly addition of PTP Instances to a domain. Except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, Signaling messages, or management messages that are a required response to a message received from the applicable management mechanism, a PTP Port in this state shall not transmit any PTP related messages.
PRE_MASTER	The PTP Port shall behave in all respects as though it were in the MASTER state except that it shall not transmit any PTP related messages except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, Signaling, or management messages that are a required response to a message received from the applicable management mechanism. Any changes to data sets as a result of the applicable management mechanism shall be atomic with respect to any other activity accessing the data sets (see 9.6).
MASTER	The PTP Port is behaving as a PTP Port in the MASTER state. It shall transmit any PTP message required by the protocol for the MASTER state or any required response to a message received from the applicable management mechanism. Any changes to data sets as a result of the applicable management mechanism shall be atomic with respect to any other activity accessing the data sets (see 9.6).
PASSIVE	Except for Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, Signaling messages, or management messages that are a required response to a message received from the applicable management mechanism, a PTP Port in this state shall not transmit any PTP related messages. Any changes to data sets as a result of the applicable management mechanism shall be atomic with respect to any other activity accessing the data sets (see 9.6).
UNCALIBRATED	One or more PTP Ports in the MASTER state have been detected in the domain. The appropriate PTP Port in the MASTER state has been selected, and the local PTP Port is preparing to perform clock adjustments with respect to the selected PTP Port in the MASTER state. This is a transient state to allow initialization of servos, updating of data sets when a new PTP Port in the MASTER state has been selected, and other implementation-specific activity. It shall transmit any PTP message required by the protocol for the UNCALIBRATED state or any required response to a message received from the applicable management mechanism. Any changes to data sets as a result of the applicable management mechanism shall be atomic with respect to any other activity accessing the data sets (see 9.6).
SLAVE	The PTP Port is performing clock adjustments with respect to the selected PTP Port in the MASTER state. It shall transmit any PTP message required by the protocol for the SLAVE state or any required response to a message received from the applicable management mechanism. Any changes to data sets as a result of the applicable management mechanism shall be atomic with respect to any other activity accessing the data sets (see 9.6).
NOTE—PTP-related messages include PTP messages defined by the protocol as well as responses to messages from the applicable management mechanism.	

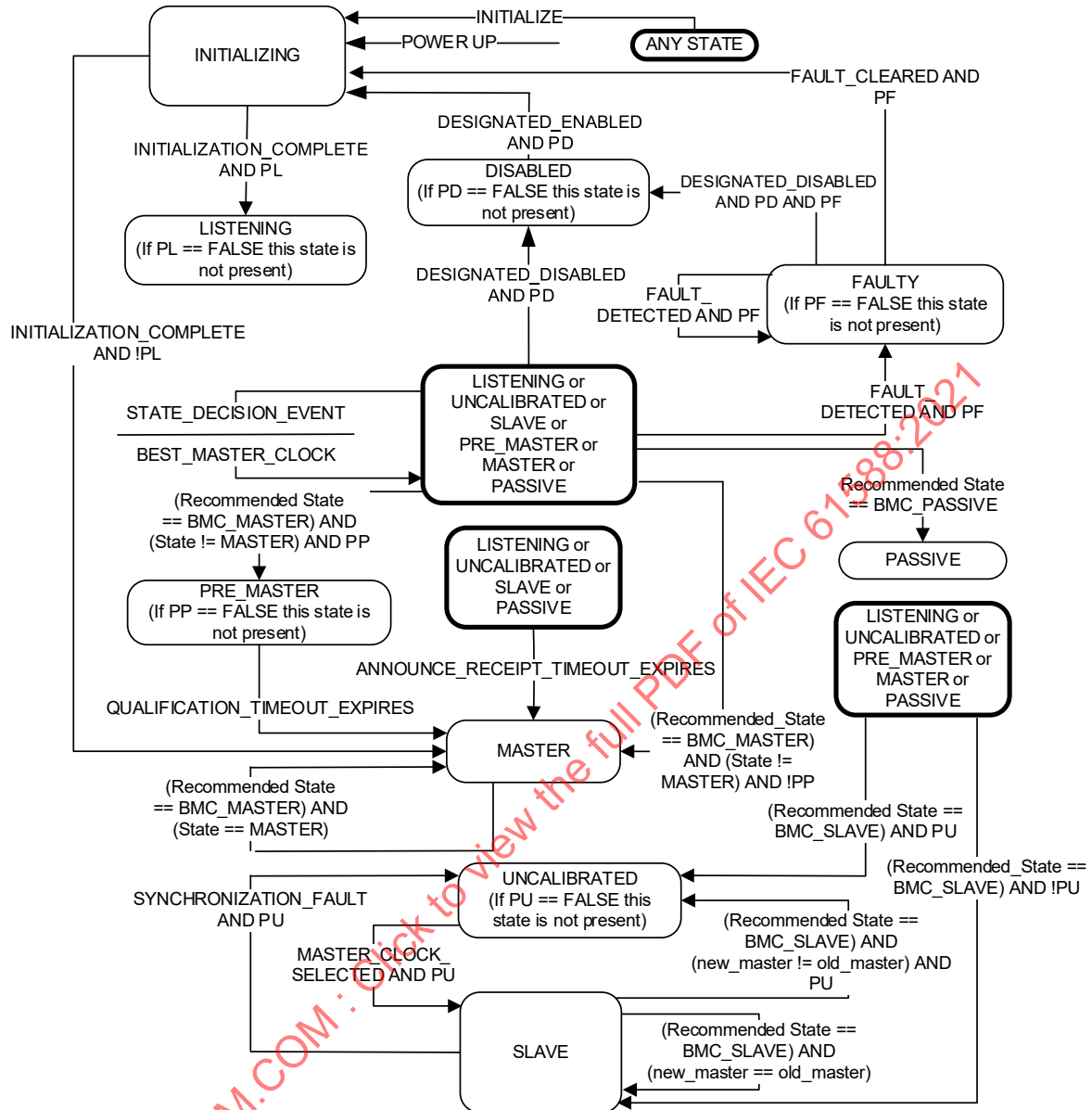


Figure 30—State machine for a full implementation

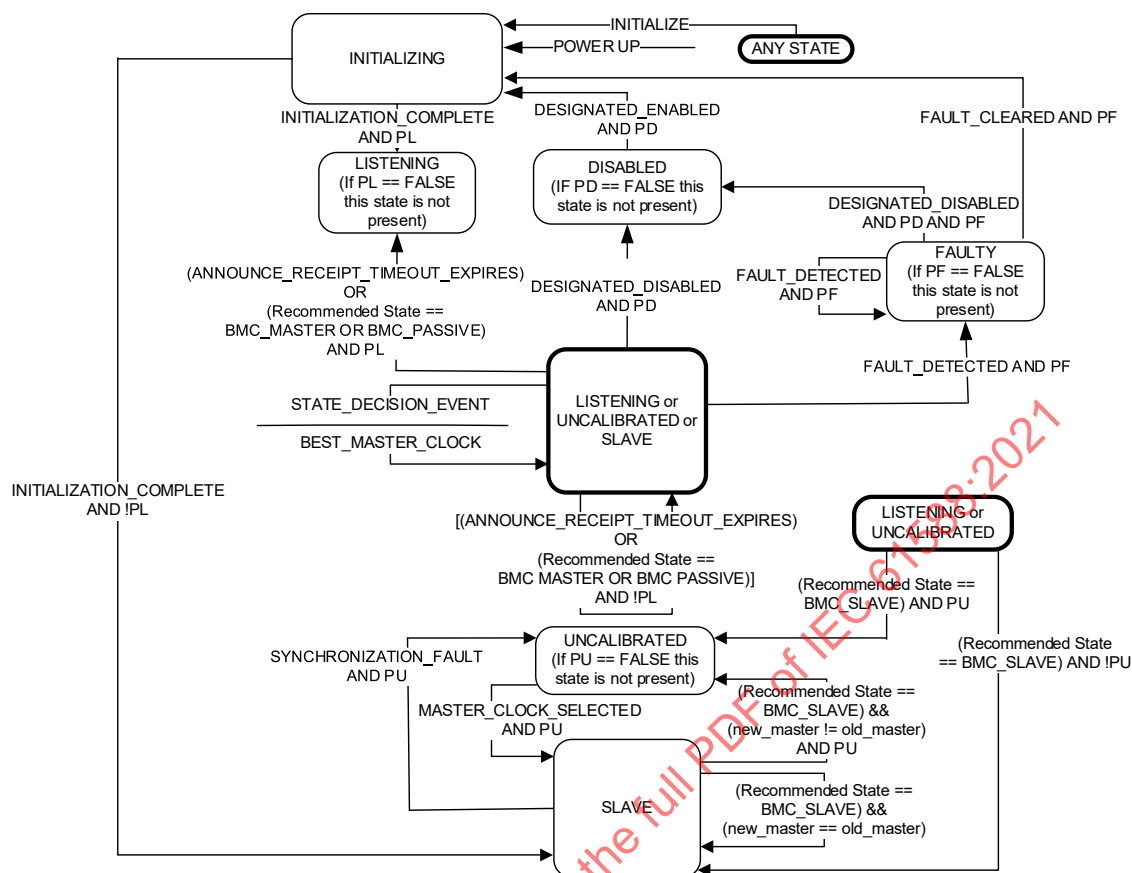


Figure 31—State machine for a slave-only implementation

In both Figure 30 and Figure 31, when a fault is cleared or a previously disabled PTP Port is enabled, the state machine makes a transition to the INITIALIZING state.

If the option of 17.7 is implemented, in Figure 30 and Figure 31, the values of PD, PF, PL, PP, and PU are set by the choices made in 17.7. If the option of 17.7 is not implemented, then these values shall be TRUE.

NOTE 1—The values of the attributes PD, PF, PL, PP, and PU determine whether the PTP Port states DISABLED, FAULTY, LISTENING, PRE-MASTER, and UNCALIBRATED, respectively, are used.

NOTE 2—The option of 17.7 fixes the values of PD, PF, PL, PP, and PU by design. Therefore, an implementation need not explicitly maintain these values and test for them in implementing the state machine logic but instead can simply note the value of each and implement the resulting logical specification of the transitions.

NOTE 3—The holdover upgrade option of 16.4 can influence the state of a PTP Port via changes in the value of clockClass.

NOTE 4—A PTP Port in the SLAVE state is expected to execute the BMCA and only accept PTP timing messages from its best master.

9.2.6 Events initiating PTP state transitions

9.2.6.1 General

The specifications for each event initiating a transition in the state machines of Figure 30 and Figure 31 are defined in 9.2.6.

9.2.6.2 POWERUP

The POWERUP event shall be instantiated by turning on the power to the PTP Node. It may also be instantiated by implementation-specific reset mechanisms, for example, a reset button.

9.2.6.3 INITIALIZE

The INITIALIZE event shall be instantiated when the instanceEnable data set member transitions from FALSE to TRUE (see 8.2.1.5.2). If management write of the instanceType is supported (see 8.2.1.5.5), the INITIALIZE event shall be instantiated when instanceType is changed.

9.2.6.4 DESIGNATED_ENABLED

The DESIGNATED_ENABLED event shall be instantiated when the portEnable data set member transitions to TRUE (see 8.2.15.5.1).

9.2.6.5 DESIGNATED_DISABLED

The DESIGNATED_DISABLED event shall be instantiated when the portEnable data set member transitions to FALSE (see 8.2.15.5.1).

9.2.6.6 FAULT_CLEARED

The FAULT_CLEARED event shall be instantiated by the clearing of the fault condition or conditions that prevents correct operation of the PTP Port.

NOTE—The clearing of all detected fault conditions can result from the actions of PTP management messages or internal procedures.

9.2.6.7 FAULT_DETECTED

The FAULT_DETECTED event shall be instantiated by the occurrence of an internal condition that prevents the correct operation of the PTP Port.

9.2.6.8 INITIALIZATION_COMPLETE

The INITIALIZATION_COMPLETE event shall be instantiated when the implementation-specific initialization process has completed.

NOTE—This was an implied event in the 2008 edition as indicated by the transaction out of the INITIALIZING state with no guard in Figures 23 and 24 of that edition. The event is explicit in this edition to allow the logic of the option of 17.7 to be expressed in Figure 30 and Figure 31.

9.2.6.9 STATE_DECISION_EVENT

The STATE_DECISION_EVENT is the mechanism for using the data in received Announce messages to determine which is the best Master PTP Instance, and whether the PTP Instance's PTP Port or PTP Ports needs to change state.

Every Ordinary Clock and Boundary Clock shall implement a mechanism generating the STATE_DECISION_EVENT, and the occurrence of this event shall implement the logic illustrated in Figure 32. In Figure 32, the best master clock algorithm is illustrated as the default best master clock algorithm specified in this standard. If an alternate best master clock algorithm is specified (see 9.3.1), the portion of the figure within the box labeled best master clock algorithm may differ.

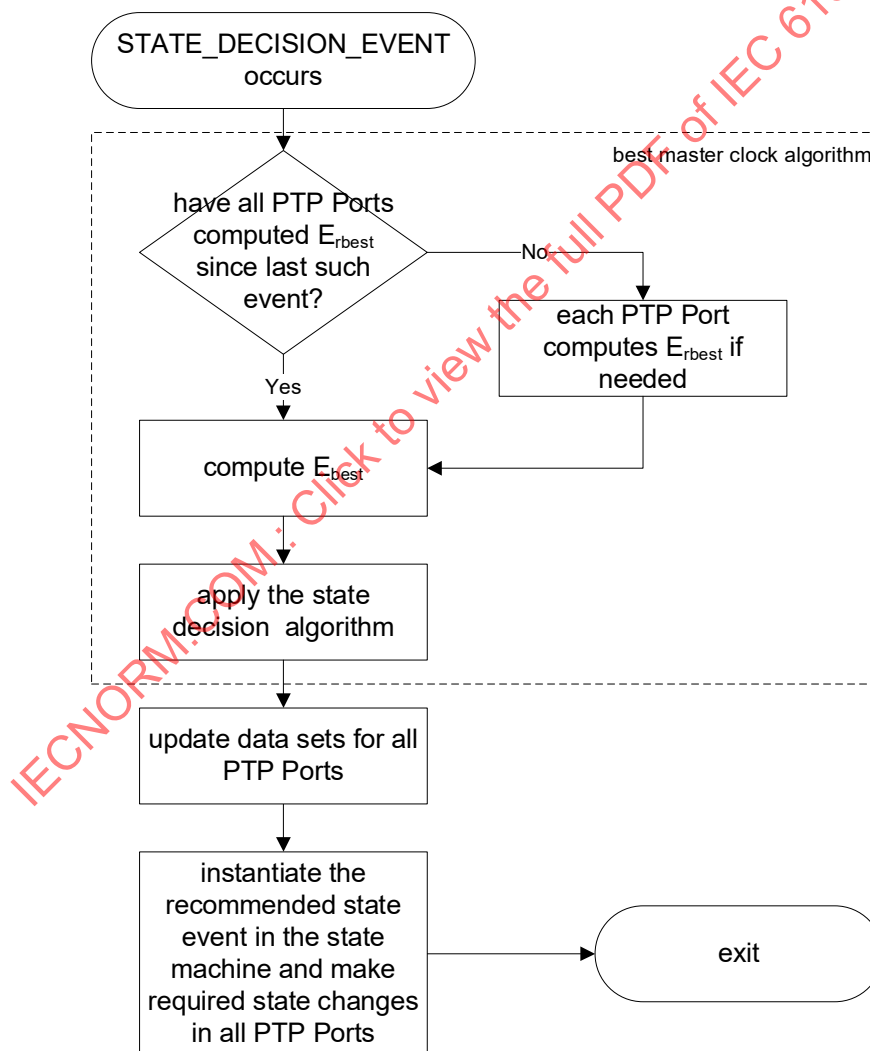


Figure 32—STATE_DECISION_EVENT logic

The STATE_DECISION_EVENT shall:

- Logically occur simultaneously on all PTP Ports of a PTP Instance
- Occur at least once per Announce message transmission interval
- Not occur when all PTP Port are in the INITIALIZING state

For PTP Instances implementing the default best master clock algorithm (see 9.3.1), prior to or as the first action of the STATE_DECISION_EVENT logic, each PTP Port not in the DISABLED, FAULTY, or INITIALIZING states shall compute an updated value of $E_{r_{best}}$ (see 9.3.2.3), reflecting the receipt of Announce messages since the last STATE_DECISION_EVENT. Following the computation of the set of $E_{r_{best}}$ for all PTP Ports, the PTP Instance shall compute E_{best} .

All Ordinary Clocks and Boundary Clocks shall complete the following tasks, in the order given:

- Apply the best master clock algorithm.
- Update the appropriate data sets.
- Instantiate the recommended state event in the state machine.
- Make the required state changes in all PTP Ports.

These tasks shall be carried out atomically (see 3.1.2). These tasks shall be executed as defined in 9.3.

9.2.6.10 Recommended state

The recommended state event results from the exercise of the best master clock algorithm initiated by a STATE_DECISION_EVENT.

9.2.6.11 QUALIFICATION_TIMEOUT_EXPIRES

The expiration of the `<qualificationTimeoutInterval>` defines the QUALIFICATION_TIMEOUT_EXPIRES event. This timeout mechanism determines the interval a PTP Instance spends in the PRE_MASTER state.

The `<qualificationTimeoutInterval>` shall start when the PTP Port enters the PRE_MASTER state. The expiration shall occur after an interval computed as follows:

The `<qualificationTimeoutInterval>` shall be N multiplied by the `announceInterval` (see 7.7.2.2), in seconds, when:

- a) If the recommended state = MASTER event was based on decision points M1 or M2 of Figure 33, N shall be 0.
- b) If the recommended state = MASTER event was based on decision point M3 of Figure 33, N shall be the value incremented by 1 (one) of the value of `currentDS.stepsRemoved` member.

9.2.6.12 ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES

Each protocol engine shall support a timeout mechanism defining the `<announceReceiptTimeoutInterval>`, with a value of `portDS.announceReceiptTimeout` multiplied by the `announceInterval` (see 7.7.3.1).

The ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES event occurs at the expiration of this timeout plus a random number uniformly distributed in the range (0,1) `announceIntervals`.

This timeout mechanism for a PTP Port shall start or be restarted when any of the following apply:

- a) For a PTP Port in the UNCALIBRATED or SLAVE states, when an Announce message is received from the Parent PTP Instance as indicated by the parentDS.parentPortIdentity
- b) At the expiration of the current <announceReceiptTimeoutInterval> unless otherwise specified in Clause 9
- c) When entering the LISTENING, UNCALIBRATED, SLAVE, or PASSIVE state
- d) For a PTP Port in the PASSIVE state when an Announce message is received from the PTP Instance that transmitted the Announce message that caused the PTP Port to be in the PASSIVE state, as indicated by a comparison of the sourcePortIdentity fields of the respective messages

This timeout mechanism for a PTP Port shall be stopped and not restarted when entering the INITIALIZING, PRE_MASTER, FAULTY, DISABLED, or MASTER states.

In addition to the state changes of Figure 30 and Figure 31, PTP Ports in the LISTENING, PASSIVE, UNCALIBRATED, or SLAVE states when the ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES event occurs shall update data sets prior to entering the MASTER state as follows:

- For an Ordinary Clock, update the PTP Port's data sets to the MASTER state configuration (see 9.3.5), as specified for decision M1.
- For a Boundary Clock with no other PTP Port in the SLAVE state, update the PTP Port's data sets to the MASTER state configuration (see 9.3.5), as specified for decision M1.
- For a Boundary Clock with a different PTP Port in the SLAVE state, update the PTP Port's data sets to the MASTER state configuration (see 9.3.5) as specified for decision M3.

9.2.6.13 SYNCHRONIZATION_FAULT

The SYNCHRONIZATION_FAULT event instantiation is implementation specific. This event should be instantiated whenever a PTP Port is in the SLAVE state and the implementation detects implementation circumstances that require re-execution of functions that occur in the UNCALIBRATED state to ensure correct synchronization.

9.2.6.14 MASTER_CLOCK_SELECTED

MASTER_CLOCK_SELECTED event instantiation is implementation specific. This event should be instantiated whenever a PTP Port in the UNCALIBRATED state has satisfied all implementation- and protocol-mandated requirements necessary to ensure synchronization when in the SLAVE state.

9.2.7 Applying PTP events to the PTP Ports of a Boundary Clock

For a Boundary Clock, the events initiating a transition in the state machines of Figure 30 and Figure 31 (see 9.2.6) shall be applied to the PTP Instance's PTP Port state machines as specified in Table 28.

Table 28—Event applicability in Boundary Clocks

Event name	Port applicability
POWERUP	All PTP Ports.
INITIALIZE	All PTP Ports.
FAULT DETECTED	All PTP Ports affected by the fault.
FAULT CLEARED	All PTP Ports affected by the fault.
STATE DECISION EVENT	All PTP Ports.
Recommended state, see NOTE	All PTP Ports.
ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES or QUALIFICATION_TIMEOUT_EXPIRES	The PTP Port signaling the expiring timeout mechanism.
DESIGNATED_ENABLED or DESIGNATED_DISABLED	The PTP Ports specified by the initiating PTP management mechanism.
MASTER_CLOCK_SELECTED	The PTP Port signaling the event.
SYNCHRONIZATION_FAULT	The PTP Port signaling the event.
NOTE—Recommended state is a condition resulting from the exercise of the best master clock algorithm initiated by a STATE DECISION EVENT.	

9.3 Best master clock algorithms

9.3.1 Selection of the best master clock algorithm

PTP permits the use of two forms of best master clock algorithm.

- By default, the mechanism specified in 9.3.2, 9.3.3, and 9.3.4, or
- If specified in the applicable PTP Profile, an alternate best master clock algorithm

Any alternate best master clock algorithm shall meet the following requirements:

- a) The output of the algorithm shall provide the recommended state required for operation of the PTP state machines and state decision events of 9.2.5, 9.2.6.9, and 9.2.6.10. The recommended states shall meet the requirements of 9.2.4. Optionally, the alternate algorithm may be a dynamic algorithm, or it may be a static algorithm that simply configures the recommended state values on the PTP Ports of the PTP Instance on which it is running.
- b) The output of the algorithm shall provide the state decision codes for use in updating data sets (see 9.3.5) and any data required for implementing the updates based on these codes. These decision codes shall be as follows:
 - 1) **M1:** The PTP Port is in the MASTER state because it is on a clockClass 1 through 127 PTP Instance and is a PTP Port of the Grandmaster PTP Instance of the domain.
 - 2) **M2:** The PTP Port is in the MASTER state because it is on a clockClass 128 or higher PTP Instance and is a PTP Port of the Grandmaster PTP Instance of the domain.
 - 3) **M3:** The PTP Port is in the MASTER state, but it is not a PTP Port on the Grandmaster PTP Instance of the domain.
 - 4) **S1:** The PTP Port is in the SLAVE state.
 - 5) **P1:** The PTP Port is in the PASSIVE state because it is on a clockClass 1 through 127 PTP Instance and is either not on the Grandmaster PTP Instance of the domain or is PASSIVE to break a timing loop.
 - 6) **P2:** The PTP Port is in the PASSIVE state because it is on a clockClass 128 or higher PTP Instance and is PASSIVE to break a timing loop.

The best master clock algorithm is run locally on all PTP Ports of every Ordinary Clock and Boundary Clock in a domain. Since it runs continuously, it continually readapts to changes in the PTP Network or the PTP Instances.

9.3.2 Best master clock algorithm

9.3.2.1 Overview and definition of terms

Subclause 9.3.2 specifies the way that a PTP Instance determines which of all the PTP Instances (including itself) is the “best.” From that it determines the next state of all its PTP Ports (see Table 27). The algorithm runs independently on each PTP Instance in a domain. In other words, PTP Instances do not negotiate which should be the Master PTP Instance and which should be the Slave PTP Instance on a PTP Communication Path—instead, each computes only the state of its own PTP Ports. The algorithm avoids PTP Communication Path configurations with two PTP Instances with a PTP Port in the MASTER state, or none, or ones that thrash.

The best master clock algorithm consists of the following two parts:

- a) A data set comparison algorithm that determines which of two PTP Instances is better. This algorithm is specified in 9.3.4.
- b) An algorithm that computes a recommendation for the state of each PTP Port involved. This algorithm is specified in 9.3.3. The recommendation is called “recommended state” in Figure 30, Figure 31, and Figure 33.

This determination is based on information contained in Announce messages received at the PTP Ports of a given PTP Instance and on the defaultDS data set values of the given PTP Instance.

9.3.2.2 General best master clock algorithm specifications

The best master clock algorithm on an Ordinary Clock or Boundary Clock C_0 characterized by a defaultDS data set D_0 and other data sets, and with “N” PTP Ports, shall be as follows:

- a) For each PTP Port “r” of C_0 , qualified Announce messages (see 9.3.2.5) received from PTP Ports of other PTP Instances connected to the PTP Communication Path used by PTP Port “r” shall be compared.
- b) For each PTP Port “r” of C_0 , the best of these messages, $E_{r\text{best}}$, shall be determined using the data set comparison algorithm.
- c) For the set of N PTP Ports of PTP Instance C_0 , the best of all messages, E_{best} , shall be determined from the N $E_{r\text{best}}$ messages using the data set comparison algorithm.
- d) For each N PTP Port of PTP Instance C_0 , the messages E_{best} and $E_{r\text{best}}$ and the defaultDS data set D_0 shall be used with the state decision algorithm to determine the best master clock event applicable to the state machine (see 9.2.5) of each PTP Port.
- e) Except for the provisions of the grandmaster cluster option (see 17.2) a PTP Port shall discard all Announce messages with alternateMasterFlag TRUE in the exercise of the best master clock algorithm under the terms of 9.3.2. These messages shall not be entered into the <foreignMasterList> for consideration by the best master clock algorithm. The alternate master option (see 17.3) specifies other uses for Announce messages with this flag set to TRUE.

9.3.2.3 Computation of $E_{r\text{best}}$

Each PTP Port may determine $E_{r\text{best}}$ independently of activity on other PTP Ports. The determination of E_{best} , the application of the state decision algorithm, and the instantiation of any state changes required by the application of the results of these determinations shall be atomic (see 3.1.2).

In computing $E_{r_{best}}$ a PTP Port “r” shall:

- a) Consider only qualified Announce messages (see 9.3.2.5) received on PTP Port “r.”
- b) Include at least two qualified Announce messages from a foreign master if such messages exist. In this case, “r” shall delete from its implementation-specific `<foreignMasterList>` (see 9.3.2.4) all such records that were considered, but not selected, as $E_{r_{best}}$.

NOTE—If the option of 17.7 specifies that the foreign master function is not to be used, then there will never be Announce messages designated as a foreign master.

- c) If PTP Port “r” is in the SLAVE, UNCALIBRATED, or PASSIVE state, include the results of the previous computation of $E_{r_{best}}$ on PTP Port “r.” However, if a more recent qualified Announce message is received on PTP Port “r” from the same sending PTP Port, the values from that message shall be considered instead. If an ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES event occurs (see 9.2.6.12) for the PTP Instance selected during the previous computation of $E_{r_{best}}$ on PTP Port “r,” then the previous computation of $E_{r_{best}}$ on PTP Port “r” shall not be included.

Announce messages should be included from as many `<foreignMasterList>` entries as PTP Port resources permit.

9.3.2.4 `<foreignMasterList>` specifications

9.3.2.4.1 General

Unless otherwise specified in the option of 17.7, each PTP Port shall maintain an implementation-specific `<foreignMasterList>` for the purposes of qualifying Announce messages, and supporting the BMCA. Each entry of the `<foreignMasterList>` contains two or three members:

- `<foreignMasterList>[].foreignMasterPortIdentity`,
- `<foreignMasterList>[].foreignMasterAnnounceMessages`, and optionally
- `<foreignMasterList>[].mostRecentAnnounceMessage`.

NOTE—The option of 17.7 permits not using the `<foreignMasterList>`.

9.3.2.4.2 `<foreignMasterList>[].foreignMasterPortIdentity`

The value of `<foreignMasterList>[].foreignMasterPortIdentity` shall be the `sourcePortIdentity` field value of an Announce message received from the foreign master.

9.3.2.4.3 `<foreignMasterList>[].foreignMasterAnnounceMessages`

The value of `<foreignMasterList>[].foreignMasterAnnounceMessages` shall be the number of Announce messages from the foreign master indicated by the `<foreignMasterList>[].foreignMasterPortIdentity` member that have been received within a time window of duration `FOREIGN_MASTER_TIME_WINDOW`.

9.3.2.4.4 `<foreignMasterList>[].mostRecentAnnounceMessage`

The value of `<foreignMasterList>[].mostRecentAnnounceMessage`, if maintained, shall be the most recent Announce message received on the PTP Port maintaining the `<foreignMasterList>`, from the foreign master

identified by the associated `<foreignMasterList>[].foreignMasterPortIdentity` (i.e., the Announce message whose receipt resulted in the updates to the associated `<foreignMasterList>[].foreignMasterPortIdentity`, and/or `<foreignMasterList>[].foreignMasterAnnounceMessage` members of this entry. If an implementation does not maintain `<foreignMasterList>[].mostRecentAnnounceMessage` member as part of the `<foreignMasterList>` entries it must maintain alternative implementation-specific attributes that support other requirements of this standard (see 6.1.1), for example, to support the calculation of $E_{r_{best}}$ for the port (see 9.3.2.3).

NOTE—Part of the information contained in the Announce message (e.g., its sequence ID) is needed to determine that the Announce message is a new Announce message from the specific foreign master. Part of the information contained in the Announce message might also be required for executing the Data set comparison algorithm (see 9.3.4) during calculation of $E_{r_{best}}$ for the port (see 9.3.2.3).

9.3.2.4.5 FOREIGN_MASTER_TIME_WINDOW and FOREIGN_MASTER_THRESHOLD

The values of these two attributes determine the criteria for accepting an Announce message from a previously silent PTP Instance for consideration in the operation of the best master clock algorithm. The values of these attributes shall be:

- FOREIGN_MASTER_TIME_WINDOW: 4 announceInterval
- FOREIGN_MASTER_THRESHOLD: 2 Announce messages received within the FOREIGN_MASTER_TIME_WINDOW

NOTE—The option of 17.7 permits not using the `<foreignMasterList>`. Implementations can accomplish this by either not implementing the functions of 9.3.2.4 or by setting the value of the FOREIGN_MASTER_THRESHOLD to 0.

9.3.2.4.6 Size of `<foreignMasterList>`

The implementation-specific `<foreignMasterList>` shall have a minimum capacity of five foreign master records in the list.

9.3.2.5 Qualification of Announce messages

An Announce message S received by a PTP Port “r” shall be qualified for consideration by the best master clock algorithm as follows:

- a) If S was sent from PTP Port “r” or from any other PTP Port on the PTP Instance containing PTP Port “r” (see 9.5.2), S shall not be qualified.
- b) If S is not the Announce message most recently received on PTP Port “r” from a given PTP Instance, S shall not be qualified.
- c) Unless otherwise specified by the option of 17.7, if the sender of S is a foreign master F, and fewer than FOREIGN_MASTER_THRESHOLD distinct Announce messages from F have been received within the most recent FOREIGN_MASTER_TIME_WINDOW interval, S shall not be qualified. Distinct Announce messages are those that have different sequenceIds, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field.

NOTE 1—If the option of 17.7 has determined that the foreign master feature is not to be used and this is implemented by setting the FOREIGN_MASTER_THRESHOLD to 0, then the operation of this specification, that is, in this provision “c”, would not result in the PTP message being disqualified. If the implementation simply does not implement the foreign master function, then specification “c” is nonoperative.

NOTE 2—The purpose of this window and threshold is to qualify only stable potential masters and prevent spurious transitions between Master PTP Instances.

- d) If the stepsRemoved field of S is 255 or greater, S shall not be qualified.

NOTE 3—This provision ensures that rogue Announce messages are extinguished. This is a mandatory backup to the use of the PATH_TRACE option for this purpose (see 16.2). This stepsRemoved-based mechanism can cause the failure of PTP if the size of the PTP Network is such that there are paths through the PTP Network involving 255 Boundary Clocks. This is extremely unlikely in practical applications.

- e) This specification “e” is optional. If the option is used, then if the stepsRemoved field of S is greater than or equal to the value of defaultDS.maxStepsRemoved, S shall not be qualified. The value of defaultDS.maxStepsRemoved may be set by a PTP Profile or network designers to help decrease the delay in eliminating rogue Announce messages. The value need not be the same for each PTP Instance in the domain. Care should be taken to ensure that selected values are set high enough to enable qualification of PTP messages arriving along the longest allowed path between any PTP Instance and its possible Grandmaster PTP Instances. See Annex K for a discussion of suppression of rogue Announce messages and the selection of appropriate values for defaultDS.maxStepsRemoved.

NOTE 4—This provision “e” allows configuration to ensure that rogue Announce messages are extinguished sooner than would be the case with the 255 limit of the previous item. Performance can be improved by setting the values of defaultDS.maxStepsRemoved to values appropriate to the actual PTP Network topology.

NOTE 5—9.3.2.5 is part of the definition of the best master clock algorithm. Profiles specifying alternate BMCAs under the provisions of 9.3.1 that also want to include the protection against rogue Announce messages enabled by the option of item “e” or the specification of item “d” of 9.3.2.5 need to include these specifications in the profile-defined alternate BMCA.

- f) Otherwise, S shall be qualified.

9.3.3 State decision algorithm

The state decision algorithm used in the best master clock algorithm shall be as defined by Figure 33. After a decision is reached by use of this algorithm, the data sets of the PTP Instance shall be updated as specified in 9.3.5.

D_0 represents the characteristics of PTP Instance C_0 as specified in the data sets of C_0 .

Comparisons of D_0 with E_{best} or E_{rbest} shall be made using the data set comparison algorithm. For the decision block in Figure 33, “ E_{best} better by topology than E_{rbest} ,” comparisons of E_{best} with E_{rbest} shall be made using the data set comparison algorithm. The “Yes” branch out of this block is taken if E_{best} is better by topology than E_{rbest} ; otherwise, the “No” branch is taken.

The determination of whether D_0 is clockClass 1 through 127 shall be made based on the value of the defaultDS.clockQuality.clockClass member of D_0 .

The values for recommended state used in the protocol engine state machines of Figure 30 and Figure 31 shall be the values for recommended state given in Figure 33.

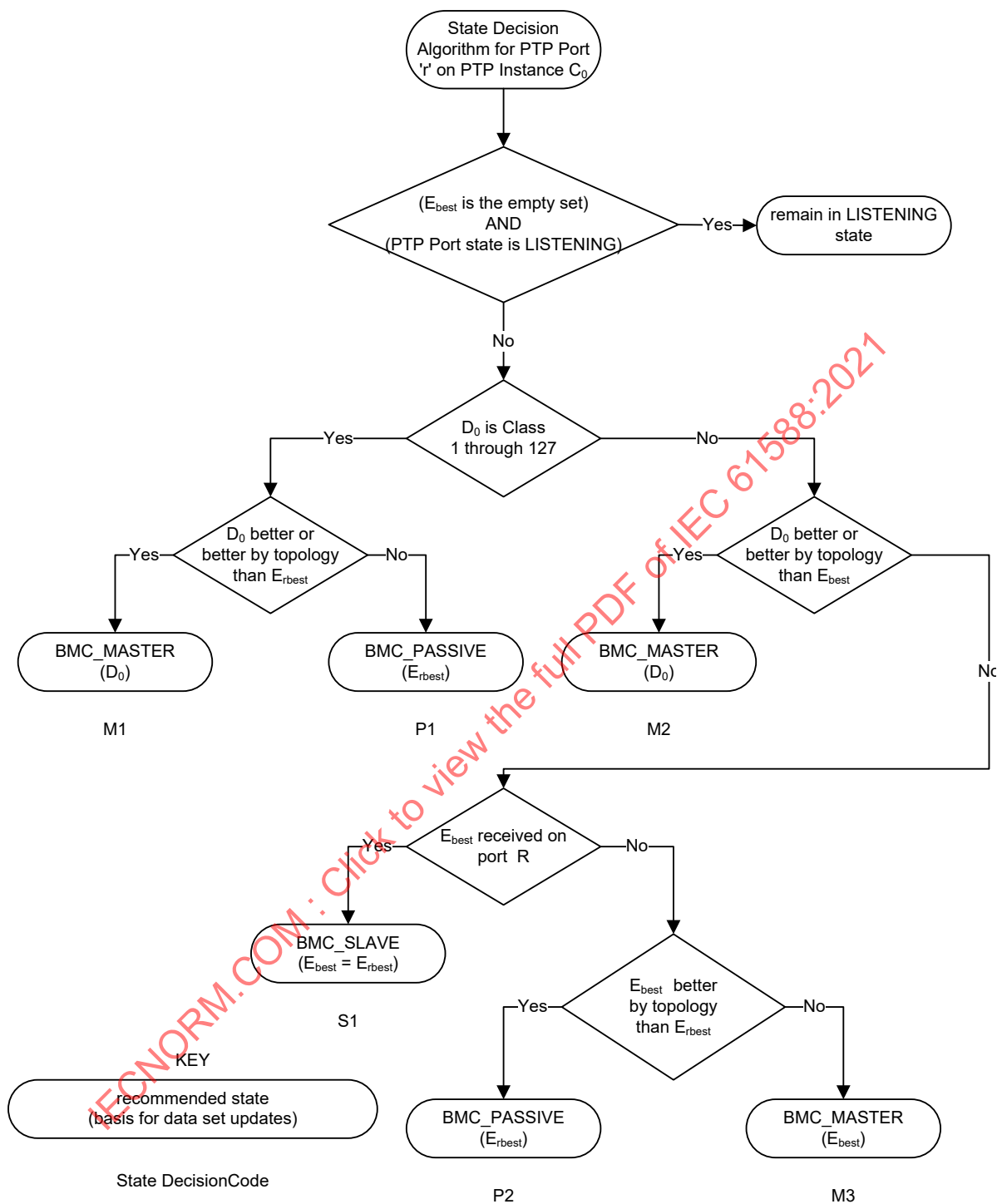


Figure 33—State decision algorithm

9.3.4 Data set comparison algorithm

The best master clock algorithm compares one PTP Instance with another by comparing data sets that represent those PTP Instances. The data set comparison algorithm shall be as defined by Figure 34 and Figure 35. The data sets are indicated in these figures as set A and set B. The sources for data set values are

given in Table 29. If in comparing any two of the data sets D_0 , E_{best} , or E_{rbest} , one of the data sets is the empty set, then the nonempty set is deemed the better set. D_0 is never the empty set; however, E_{best} or E_{rbest} or both may be the empty set.

NOTE 1—The general process followed in this comparison is as follows:

- Find which PTP Instance derives its time from the better Grandmaster PTP Instance. Choosing this, rather than which is the better PTP Instance, is essential for the stability of the algorithm.
- If those properties are equivalent, use tie-breaking techniques.

Table 29—Information sources for data set comparison algorithm

When considering this property as designated in Figure 34 and Figure 35	If the data set is E_{best} or E_{rbest} , use these fields of the associated Announce message	If the data set is D_0 , use these fields of the PTP Instance defaultDS data set
GM priority1	grandmasterPriority1	defaultDS.priority1
GM identity	grandmasterIdentity	defaultDS.clockIdentity
GM class	grandmasterClockQuality.clockClass	defaultDS.clockQuality.clockClass
GM accuracy	grandmasterClockQuality.clockAccuracy	defaultDS.clockQuality.clockAccuracy
GM offsetScaledLogVariance	grandmasterClockQuality.offsetScaledLogVariance	defaultDS.clockQuality.offsetScaledLogVariance
GM priority2	grandmasterPriority2	defaultDS.priority2
Steps Removed	stepsRemoved	The value 0
Identity of Senders	sourcePortIdentity	defaultDS.clockIdentity
Identity of Receiver	portDS.portIdentity (of portDS data set of PTP Port receiving message)	defaultDS.clockIdentity
Port Number of Receiver	portDS.portIdentity.portNumber (of portDS data set of PTP Port receiving message)	The value 0

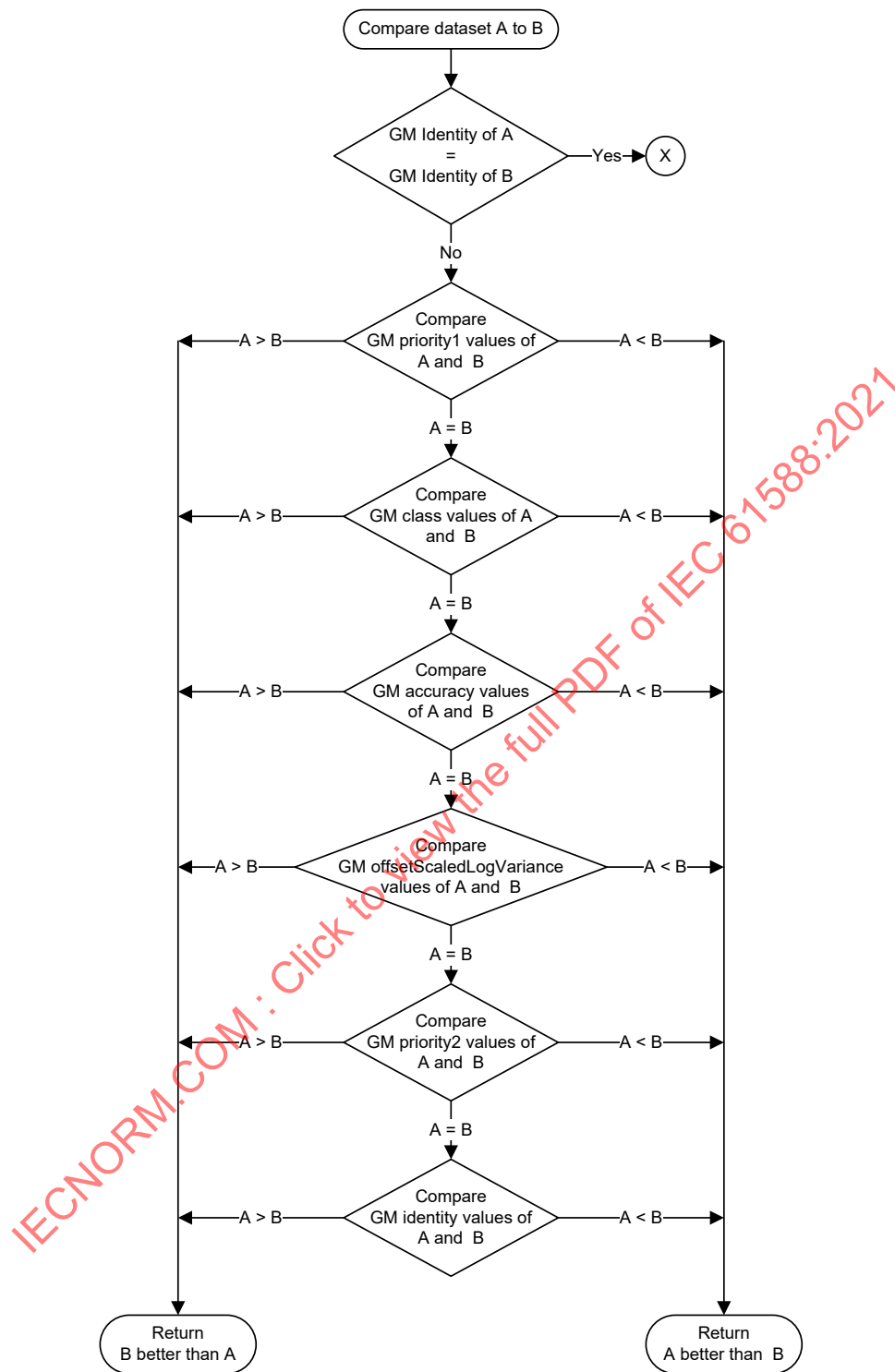


Figure 34—Data set comparison algorithm, part 1

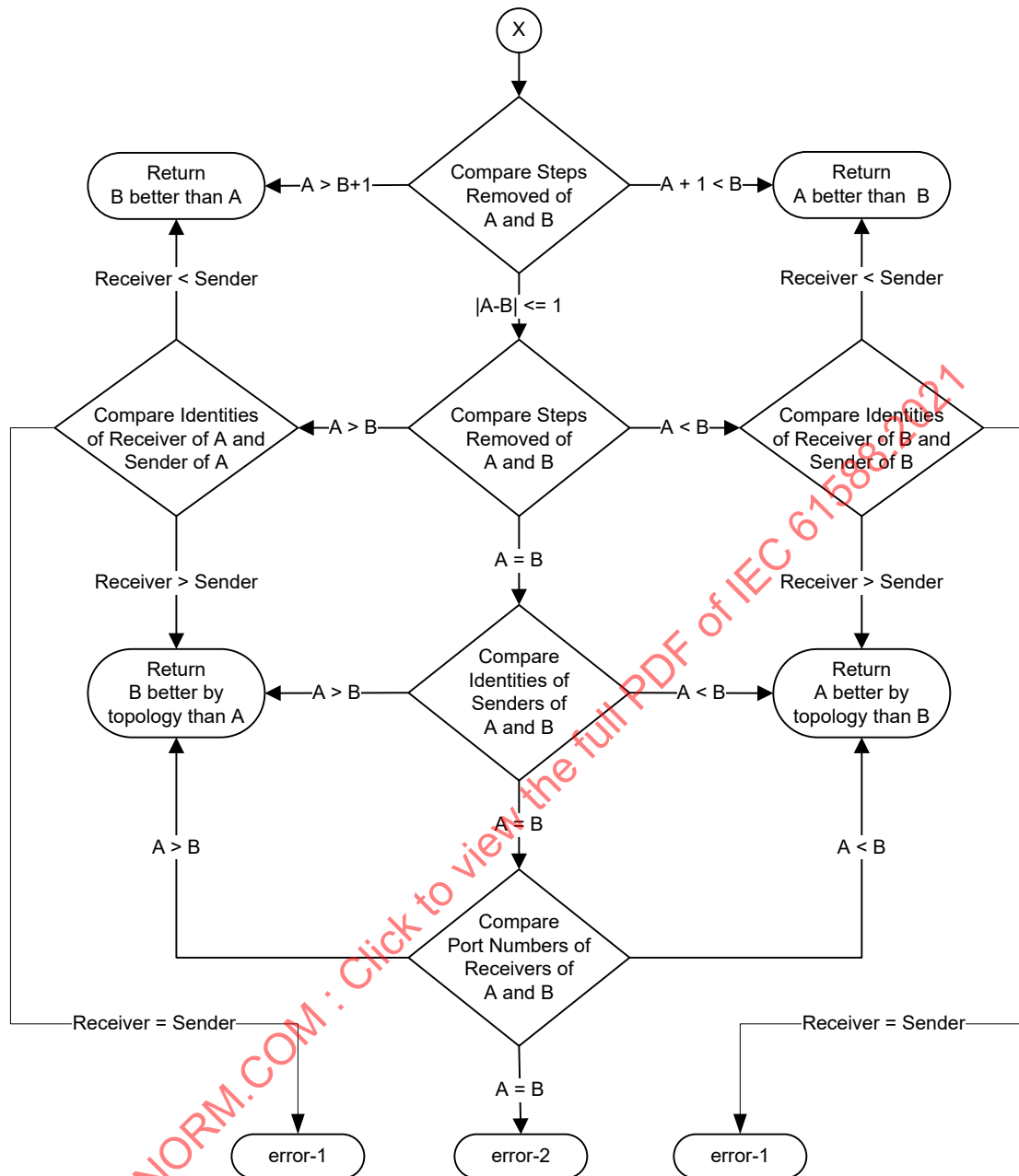


Figure 35—Data set comparison algorithm, part 2

NOTE 2—The error returns shown in Figure 35 are not used in the state decision algorithm of Figure 33. These returns are precluded by the terms of 9.3.2.2 and 9.3.2.5 but can be useful for diagnostic or error detection. The error conditions occur when terms of that subclause are violated. Error-1 indicates that one of the PTP messages was transmitted and received on the same PTP Port. Error-2 indicates that the PTP messages are duplicates or that they are an earlier and later PTP message from the same Grandmaster PTP Instance.

9.3.5 Update of data sets

The update of the currentDS, parentDS, portDS, and timePropertiesDS data sets shall be as defined in Table 30 through Table 33 for the state decision codes, which are indicated by “State Decision Codes” (i.e., M1, M2, M3, S1, P1, and P2) in Figure 33 for the default best master clock algorithm and by 9.3.1 for a PTP Profile-defined alternate.

Member portDS.portState of the portDS data set shall be updated as changes are made by the protocol state machine of Figure 30 and Figure 31 associated with each PTP Port in accordance with 9.2.6.9.

Data set fields that are not included in Table 30 through Table 33 are not updated.

Table 30—Updates for state decision code M1 and M2

Update this field	From the indicated field of the defaultDS data set of the PTP Instance unless otherwise stated
currentDS data set	
currentDS.stepsRemoved	set to 0
currentDS.offsetFromMaster	set to 0
currentDS.meanDelay	set to 0
currentDS.synchronizationUncertain	Follow rules in 8.2.2.5
parentDS.synchronizationUncertain	Follow rules in 8.2.3.11
parentDS data set	
parentDS.parentPortIdentity	parentDS.clockIdentity member set to the value of defaultDS.clockIdentity field. parentDS.parentPortIdentity.portNumber member is 0
parentDS.grandmasterIdentity	defaultDS.clockIdentity
parentDS.grandmasterClockQuality	defaultDS.clockQuality
parentDS.grandmasterPriority1	defaultDS.priority1
parentDS.grandmasterPriority2	defaultDS.priority2
timePropertiesDS data set	
timePropertiesDS.currentUtcOffset	Follow rules in 9.4
timePropertiesDS.currentUtcOffsetValid	Follow rules in 9.4
timePropertiesDS.leap59	Follow rules in 9.4
timePropertiesDS.leap61	Follow rules in 9.4
timePropertiesDS.timeTraceable	Follow rules in 9.4
timePropertiesDS.frequencyTraceable	Follow rules in 9.4
timePropertiesDS.ptpTimescale	Follow rules in 9.4
timePropertiesDS.timeSource	Follow rules in 9.4
portDS Data Set	
portDS.portState	State resulting from the application of the recommended state from Figure 33 to the state machine of Figure 30 or Figure 31 as appropriate

Table 31—Updates for state decision code M3

Update this field	From the indicated source
currentDs data set	
portDS data set	
portDS.portState	State resulting from the application of the recommended state from Figure 33 to the state machine of Figure 30 or Figure 31 as appropriate

Table 32—Updates for state decision code P1 and P2

Update this field	From the indicated source
portDS data set	
portDS.portState	State resulting from the application of the recommended state from Figure 33 to the state machine of Figure 30 or Figure 31 as appropriate

Table 33—Updates for state decision code S1

Update this field	From the indicated source
currentDS data set	
currentDS.stepsRemoved	1 + value of stepsRemoved of E _{best}
currentDS.synchronizationUncertain	Follow rules in 8.2.2.5
parentDS.synchronizationUncertain	Follow rules in 8.2.3.11
parentDS data set	
parentDS.parentPortIdentity	sourcePortIdentity of E _{best}
parentDS.grandmasterIdentity	grandmasterIdentity of E _{best}
parentDS.grandmasterClockQuality	grandmasterClockQuality of E _{best}
parentDS.grandmasterPriority1	grandmasterPriority1 of E _{best}
parentDS.grandmasterPriority2	grandmasterPriority2 of E _{best}
timePropertiesDS data set	
timePropertiesDS.currentUtcOffset	currentUtcOffset field of E _{best}
timePropertiesDS.currentUtcOffsetValid	The logical value of the currentUtcOffsetValid bit of the flagField of E _{best}
timePropertiesDS.leap59	The logical value of the leap59 bit of the flagField of E _{best}
timePropertiesDS.leap61	The logical value of the leap61 bit of the flagField of E _{best}
timePropertiesDS.timeTraceable	The logical value of the timeTraceable bit of the flagField of E _{best}
timePropertiesDS.frequencyTraceable	The logical value of the frequencyTraceable bit of the flagField of E _{best}
timePropertiesDS.ptpTimescale	The logical value of the ptpTimescale bit of the flagField of E _{best}
timePropertiesDS.timeSource	The value of the timeSource field of E _{best}
portDS data set	
portDS.portState	State resulting from the application of the recommended state from Figure 33 to the state machine of Figure 30 or Figure 31 as appropriate
NOTE—E _{best} is an Announce message.	

9.4 Grandmaster PTP Instance timePropertiesDS updates

The Grandmaster PTP Instance, 3.1.18, determines the timescale and related timescale properties of the domain.

The following specifications determine the update properties of several members of the timePropertiesDS data set of the Grandmaster PTP Instance. These properties shall be updated:

- Due to state changes mandated by Table 30 when 17.6 is not in effect, or
- Due to state changes mandated by Table 136 when 17.6 is in effect

NOTE 1—Leap-second events and the current value of UTC-TAI are posted well in advance in IERS Bulletin C; see Clause 2.

NOTE 2—The initialization specifications for these attributes are found in Clause 8.

If the timescale is PTP, the timePropertiesDS data set members shall be set as follows:

- timePropertiesDS.leap59 and timePropertiesDS.leap61: if known, to a value traceable to a primary reference, else to FALSE
- timePropertiesDS.currentUtcOffset: if known, to a value traceable to a primary reference that provides UTC, else to the value of <dLS> (see 7.2.4) when the PTP Node is designed
- timePropertiesDS.currentUtcOffsetValid: to TRUE if the value of the timePropertiesDS.currentUtcOffset and the values of the timePropertiesDS.leap59 and timePropertiesDS.leap61 are based on values obtained from the primary reference providing UTC, else to FALSE
- timePropertiesDS.ptpTimescale: to TRUE
- timePropertiesDS.timeTraceable: to TRUE if the time is traceable to a primary reference, else to FALSE
- timePropertiesDS.frequencyTraceable: to TRUE if the frequency is traceable to a primary reference, else to FALSE
- timePropertiesDS.timeSource: if known, to the appropriate value from Table 6, else to INTERNAL_OSCILLATOR

If the timescale is ARB, the timePropertiesDS data set members shall be set as follows:

- timePropertiesDS.leap59 and timePropertiesDS.leap61: to FALSE
- timePropertiesDS.currentUtcOffset: to the value of <dLS>, when the PTP Node is designed
- timePropertiesDS.currentUtcOffsetValid: to FALSE
- timePropertiesDS.ptpTimescale: to FALSE
- timePropertiesDS.timeTraceable: to TRUE if the time traceable to a primary reference, else to FALSE
- timePropertiesDS.frequencyTraceable: to TRUE if the frequency is traceable to a primary reference, else to FALSE
- timePropertiesDS.timeSource: if known, to the appropriate value from Table 6, else to INTERNAL_OSCILLATOR

If either timePropertiesDS.leap59 or timePropertiesDS.leap61 is required to be TRUE, it shall be set to TRUE during the interval between the later of the times:

- When the Grandmaster PTP Instance determined that the value is TRUE,
- 12 h prior to midnight (UTC) of the day in which the leap-second event is to occur,

and at the end of the UTC day in which the leap-second event is to occur.

The values of the timePropertiesDS.leap59 and timePropertiesDS.leap61 fields in Announce messages shall be set to FALSE starting with the first Announce message in which the updated value of timePropertiesDS.currentUtcOffset appears. See B.2.2 for details on the updates of these data sets.

Grandmaster PTP Instances shall update the value of timePropertiesDS.currentUtcOffset within ± 2 announceIntervals of midnight (UTC) of the day in which the leap-second occurred.

NOTE 3—In practical implementations, the actual update of currentUtcOffset and the setting of timePropertiesDS.leap59 or timePropertiesDS.leap61 to FALSE might occur slightly before or after midnight (UTC) due to variations in code execution time. The requirement of the ± 2 announceInterval update is easily implemented in masters and ensures, even in very large topologies, that the information is propagated to all slaves prior to noon of the following UTC day. This avoids confusion as to which midnight the indicated jump in UTC is to occur.

NOTE 4—The standard defines the propagation of the parameters `currentUtcOffset`, `currentUtcOffsetValid`, `leap59`, and `leap61` from the Grandmaster PTP Instance to other PTP Instances in the domain (see Table 30 and Table 33).

Grandmaster PTP Instances that use a recognized standard time source should distribute the timescale PTP.

9.5 PTP message processing semantics

9.5.1 Messaging behavior of Ordinary Clock and Boundary Clocks

Subclause 9.5 specifies the state and timing dependent behavior for the following PTP message related events that may occur in an Ordinary Clock, or for each PTP Port of a Boundary Clock:

- Receipt of a PTP message
- Transmission of a PTP message

The behavior resulting from these events may depend on the current state of the PTP Port of an Ordinary Clock or Boundary Clock as specified in 9.2.5.

Except for PTP management messages, all multicast PTP messages shall terminate at Boundary Clocks and Ordinary Clocks.

A Boundary Clock that is also a bridge or router forwards all unicast PTP messages as specified in 7.3.1. Any other behavior is out of the scope of this standard.

Execution of the protocol shall not result in any communication on the PTP Communication Paths, except as specified in 9.5.

Any state or data changes or data set updates of the PTP Instance resulting from the occurrence of any event or qualifying sequence of events shall be atomic (see 3.1.2). A qualifying sequence of events is defined to be the receipt of multiple Announce messages within the same `announceInterval`.

Requests to issue a PTP message resulting from state or data change or the occurrence of an event shall be processed in first-in-first-out (FIFO) order by PTP message class. A logically separate FIFO ordering shall be maintained for PTP event and PTP general messages. Once a PTP message request is queued, the PTP message shall be issued irrespective of any subsequent event occurrences.

Only PTP messages where the `domainNumber`, `majorSdoId`, and `minorSdoId` fields meet the applicable requirements of 7.1.2.1 and 7.1.3.1 are accepted for processing by the protocol.

9.5.2 Receipt by a PTP Instance of any message from itself

9.5.2.1 General

The Precision Time Protocol makes no use of PTP messages received by the same PTP Instance that sent them, and implementations should prevent such PTP messages from being considered by the protocol. Subclauses 9.5.2.2 and 9.5.2.3 specify what to do when such messages are received.

9.5.2.2 Ordinary Clocks and any single PTP Port of a Boundary Clock

A PTP message received at the same PTP Port that issued the message shall be ignored. An exception may be made for implementation-specific diagnostic purposes beyond the scope of this standard.

This situation may be identified by comparing the sourcePortIdentity field members of the received PTP message and the corresponding members of the portDS.portIdentity field of the ingress PTP Port. The possibilities and interpretations are shown in Table 34.

NOTE—This condition can occur due to normal or abnormal properties of the network segment connected to the PTP Port.

9.5.2.3 Additional constraints for Boundary Clocks

A PTP Port “N” on a Boundary Clock might directly receive Announce messages issued by a different PTP Port “M” of the same Boundary Clock. This occurs if both PTP Ports N and M communicate to the same communication path. This is an abnormal situation not detected by the best master clock algorithm. When this condition is detected in a set of the PTP Ports, the Boundary Clock shall place all of the involved PTP Ports except the PTP Port with the lowest portNumber, say N, in the PASSIVE state until such time as the normal operation of the protocol no longer determines that PTP Port N is to be in the MASTER state.

This situation may be identified by comparing the sourcePortIdentity field members of the received PTP message and the corresponding members of the portDS.portIdentity field of the ingress PTP Port. The possibilities and interpretations are shown in Table 34.

Table 34—Source identity comparisons

Given a PTP message <i>m</i> arriving at PTP Port <i>n</i> of PTP Instance <i>c</i> , where the portDS.portIdentity field of <i>n</i> has members clockIdentity <i>a</i> and portNumber <i>n</i> :	
If sourcePortIdentity of <i>m</i> contains:	The interpretation is:
clockIdentity <i>a</i> , portNumber <i>n</i>	<i>m</i> was sent from PTP Port <i>n</i> on PTP Instance <i>c</i> .
clockIdentity <i>f</i> ≠ <i>a</i> , portNumber <i>n</i>	<i>m</i> was sent from a PTP Port on a PTP Instance other than <i>c</i> .
clockIdentity <i>a</i> , portNumber <i>q</i> ≠ <i>n</i>	<i>m</i> was sent from PTP Instance <i>c</i> but from a PTP Port other than <i>n</i> .

9.5.3 Receipt of an Announce message from another PTP Instance

The logic for processing an Announce message shall be as defined in Figure 36. The states indicated in this figure refer to the current state of the PTP Port receiving the Announce message.

If the option of 17.7 is not implemented, the value of FMD in Figure 36 shall be TRUE.

If the option of 17.7 is implemented, the value of FMD in Figure 36 is set as specified in 17.7.3.

If the PTP Port receiving an Announce message is in the INITIALIZING or DISABLED states, the message shall be discarded. If the PTP Port receiving an Announce message is in the FAULTY state, the message shall be discarded except for implementation-specific purposes that otherwise meet the requirements of 9.2.

If the members of the sourcePortIdentity field of the received Announce message are identical to the corresponding members of the parentDS.parentPortIdentity of the data set of the receiving PTP Instance, then the message is from the current Parent PTP Instance, that is, the Master PTP Instance to which the PTP Instance is synchronized.

If an Announce message is received on a PTP Port in the SLAVE or UNCALIBRATED state from the current Parent PTP Instance, the data sets for the ingress PTP Port shall be updated according to Table 33, except that the source of each field shall be the received Announce message rather than E_{best}.

If the option of 17.7.3 is implemented, the remainder of 9.5.3 shall be ignored.

When an Announce message from PTP Port F is received by PTP Port N from the PTP Communication Path associated with N, PTP Port F is designated as a foreign master clock if any of the following is true:

- The receiving PTP Port N is not in the SLAVE or UNCALIBRATED state.
- The receiving PTP Port N is in the SLAVE or UNCALIBRATED state and the members of the sourcePortIdentity field of the received PTP message are not all identical to the corresponding members of the parentDS.parentPortIdentity of the data set of the receiving PTP Instance.

If an Announce message is received from a foreign master, the implementation-specific <foreignMasterList> (see 9.3.2.4) of the ingress PTP Port shall be updated as follows:

- a) If the members of the sourcePortIdentity field of the received Announce message are identical to the corresponding members of a <foreignMasterList>[].foreignMasterPortIdentity record of the ingress PTP Port, then the <foreignMasterList>[].foreignMasterAnnounceMessages field of that record shall be incremented.
- b) If the members of the sourcePortIdentity field of the received Announce message are not identical to the corresponding members of a <foreignMasterList>[].foreignMasterPortIdentity of the ingress PTP Port, then a new record shall be created with the <foreignMasterList>[].foreignMasterPortIdentity field set to the value of the sourcePortIdentity field of the received Announce message and with the <foreignMasterList>[].foreignMasterAnnounceMessages field value set to 0. Implementation-specific limitations on the capacity of the <foreignMasterList> limits the number of such records. If the <foreignMasterList> is full, then the Announce message shall be discarded.

NOTE 1—The maximum number of records that can be stored in the <foreignMasterList> (see 9.3.2.4.6) does not affect the correctness of protocol operation but can affect the speed of convergence in selecting the Master PTP Instances and Slave PTP Instances. Higher capacities permit a PTP Instance to process and eliminate from consideration more foreign masters in each announceInterval rather than wait until space is available in the <foreignMasterList> for succeeding PTP messages from any remaining foreign masters.

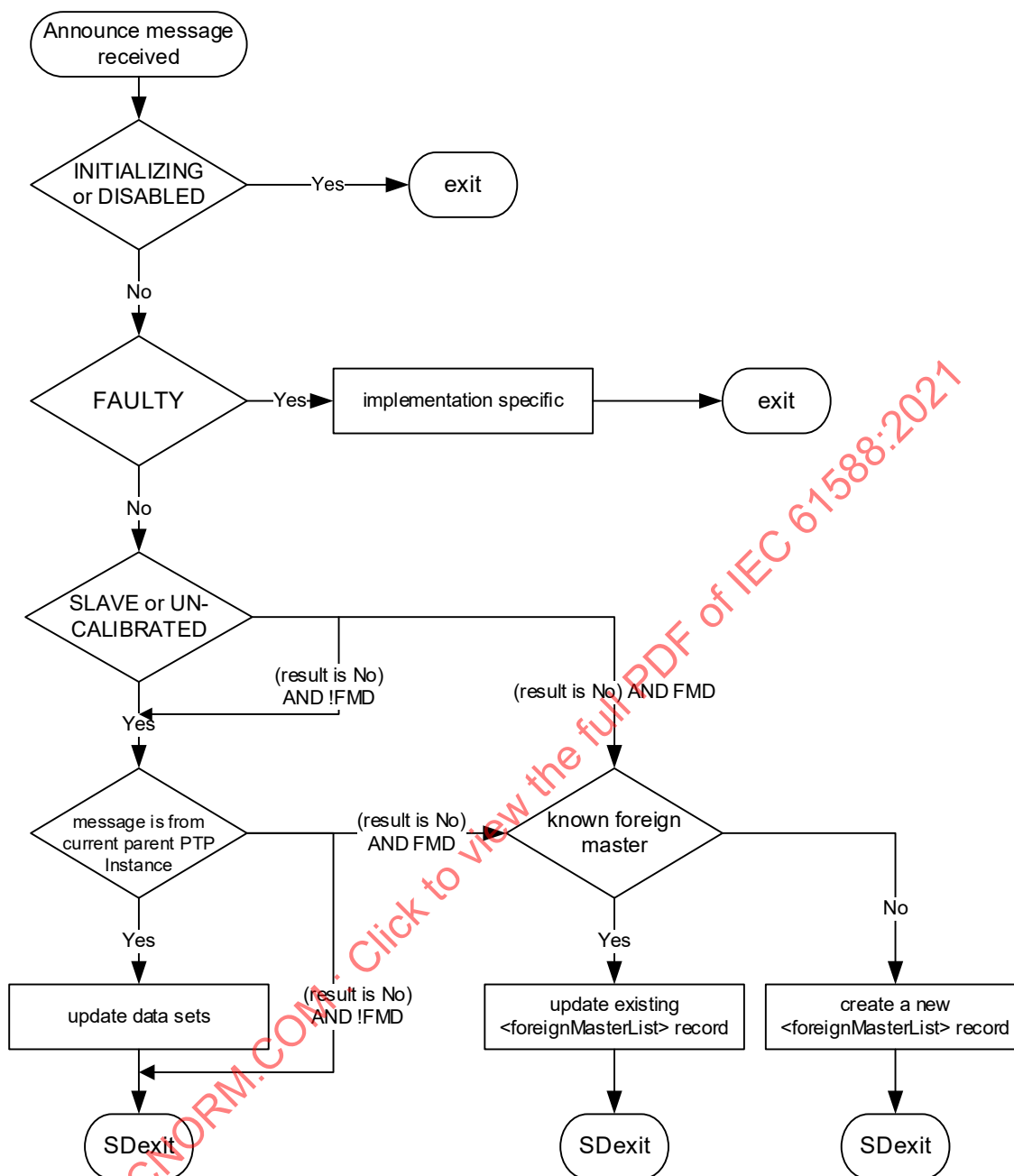


Figure 36—Receipt of Announce message logic

Upon SDexit from Figure 36:

- If the receipt of the Announce message initiates a STATE_DECISION_EVENT of 9.2.6.9, the Announce message shall be processed by the BMCA as specified in 9.3, otherwise,
- The Announce message shall be queued with other Announce messages received on the PTP Port, either in the <foreignMasterList> or queues of messages from the current master as appropriate, for processing at the next STATE_DECISION_EVENT

NOTE 2—In many implementations, the receipt of an Announce message initiates a STATE_DECISION_EVENT. In others, the STATE_DECISION_EVENT is initiated based on other criteria, for example, at the start of a new announceInterval; in which case, the BMCA will process the Announce messages queued since the last STATE_DECISION_EVENT.

9.5.4 Receipt of a Sync message from another PTP Instance

The logic for processing a Sync message shall be as defined in Figure 37. The states indicated in this figure refer to the current state of the PTP Port receiving the Sync message.

If the PTP Port receiving a Sync message is in the INITIALIZING or DISABLED states, the message shall be discarded. If the PTP Port receiving a Sync message is in the FAULTY state, the message shall be discarded except for implementation-specific purposes that otherwise meet the requirements of 9.2. If the PTP Port receiving a Sync message is not in the SLAVE or UNCALIBRATED states, the message shall be discarded.

The <syncEventIngressTimestamp> meeting the requirements of 7.3.4 shall be generated for the reception of the Sync message.

Received Sync messages should be processed as soon as possible after receipt.

If the members of the sourcePortIdentity field of the received Sync message are identical to the corresponding members of the parentDS.parentPortIdentity field of the data set of the receiving PTP Instance, then the message is from the current Parent PTP Instance.

When the Sync message is received and all the following conditions are met:

- The PTP Port receiving the Sync message is in the SLAVE or UNCALIBRATED state.
- The twoStepFlag bit of the flagField of the received Sync message is FALSE (indicating that a Follow_Up message will not be received).
- The Sync message was received from the current Parent PTP Instance.

then:

- The clock(s) of the PTP Instance shall be adjusted as defined in 12.1, based on the contents of the received Sync message, the <syncEventIngressTimestamp> and the specifications of 11.2.

When the Sync message is received and all the following conditions are met:

- The PTP Port receiving the Sync message is in the SLAVE or UNCALIBRATED state,
- The twoStepFlag bit of the flagField of the received Sync message is TRUE (indicating that a Follow_Up message will be received),
- The Sync message was received from the current Parent PTP Instance,

then:

- The PTP Port will wait for the Follow_Up message before adjusting the PTP Instance clocks as defined in 12.1 per 11.2.

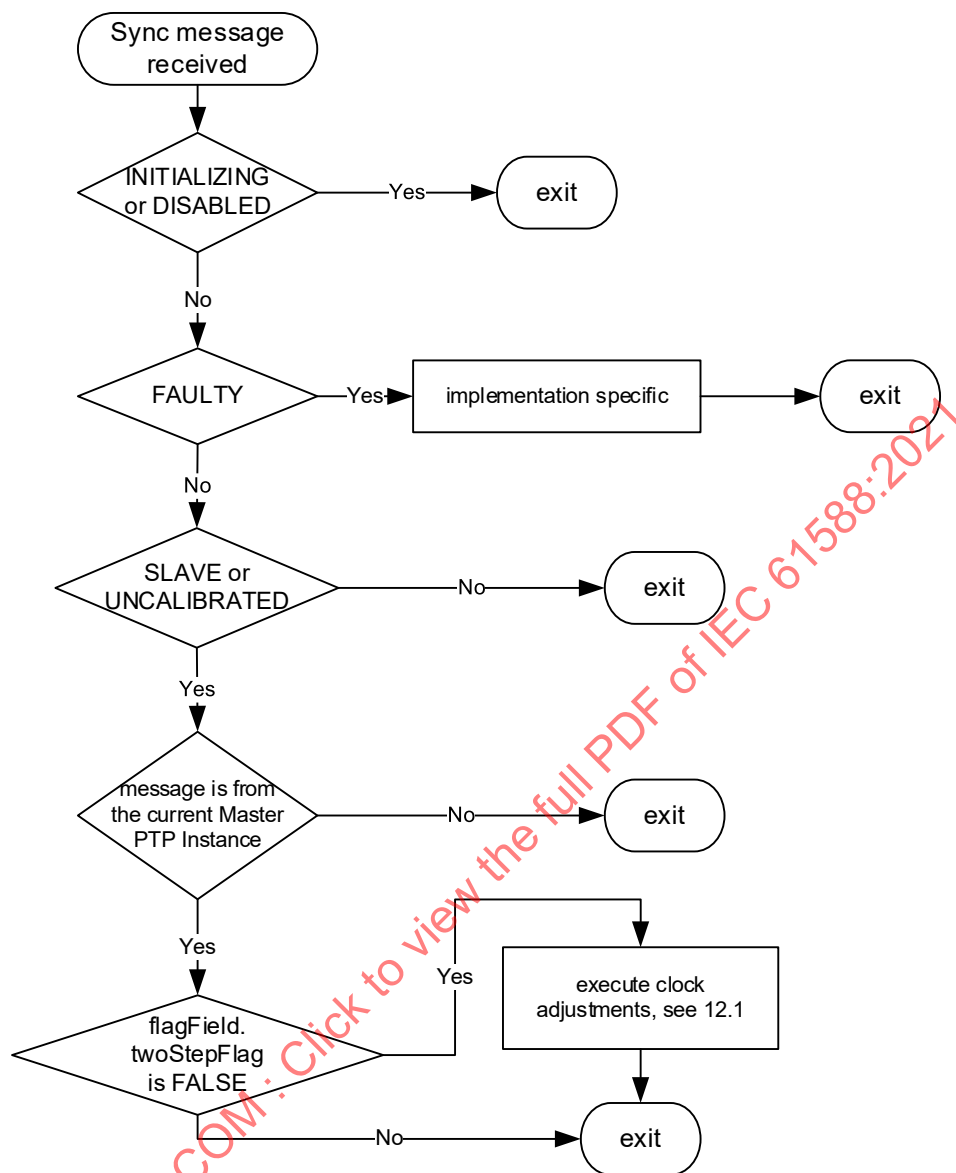


Figure 37—Receipt of Sync message logic

9.5.5 Receipt of a Follow_Up message from another PTP Instance

The logic for processing a Follow_Up message shall be as defined in Figure 38. The states indicated in this figure refer to the current state of the PTP Port receiving the Follow_Up message.

If the PTP Port receiving a Follow_Up message is in the INITIALIZING or DISABLED states, the message shall be discarded. If the PTP Port receiving a Follow_Up message is in the FAULTY state, the message shall be discarded except for implementation-specific purposes that otherwise meet the requirements of 9.2. If the PTP port receiving the Follow_Up message is not in the SLAVE nor UNCALIBRATED state, the message shall be discarded.

If the members of the sourcePortIdentity field of the Follow_Up message are identical to the corresponding members of the parentDS.parentPortIdentity member of the data set of the receiving PTP Instance, then the message is from the current Parent PTP Instance.

If the members of the sourcePortIdentity field of the received Follow_Up message are identical to the corresponding members of the sourcePortIdentity field of a prior Sync message and the sequenceId field of the received Follow_Up message matches the sequenceId field of the same prior Sync message, then the Follow_Up message and the Sync message are associated.

Follow_Up messages should be processed as soon as possible after receipt.

When the Follow_Up message is received and all of the following conditions are met:

- The PTP Port receiving the Follow_Up message is in the SLAVE or UNCALIBRATED state,
- The Follow_Up message was received from the current Parent PTP Instance,
- A prior Sync message associated with the Follow_Up message has been identified,

then the clock(s) of the PTP Instance shall be adjusted as defined in 12.1, based on the contents of the received Follow_Up message and associated Sync message, per 11.2.

9.5.6 Receipt of a Delay_Req message from another PTP Instance

The logic for processing a Delay_Req message shall be as defined in Figure 39. The states indicated in this figure refer to the current state of the PTP Port receiving the Delay_Req message.

If the PTP Port receiving the Delay_Req message is in the INITIALIZING or DISABLED states, the message shall be discarded. If the PTP Port receiving a Delay_Req message is in the FAULTY state, the message shall be discarded except for implementation-specific purposes that otherwise meet the requirements of 9.2.

Unless otherwise specified in this standard, if the PTP Port receiving the Delay_Req message is not in the MASTER state, the message shall be discarded.

The <delayReqEventIngressTimestamp> meeting the requirements of 7.3.4 shall be generated upon receipt of the Delay_Req message.

If the PTP Port receiving the Delay_Req message is in the MASTER state, the PTP Port shall transmit a Delay_Resp message subject to the conditions of 11.3 and 9.5.12.

Delay_Req messages should be processed as soon as possible after receipt.

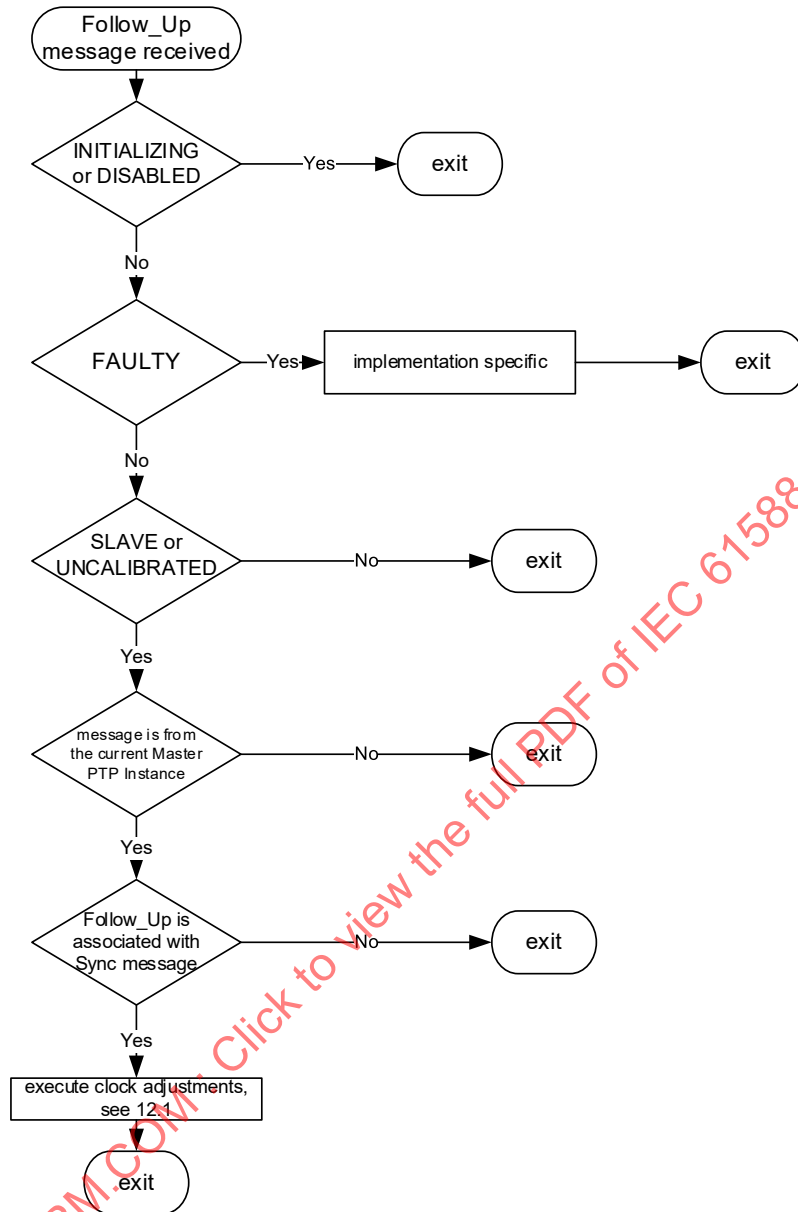


Figure 38—Receipt of Follow_Up message logic

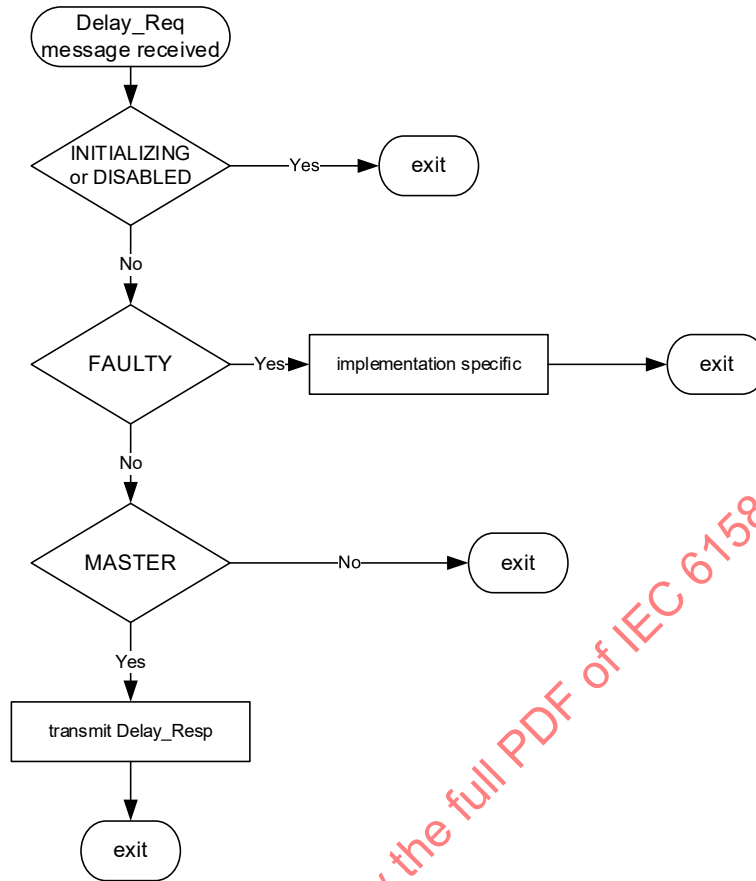


Figure 39—Receipt of Delay_Req message logic

9.5.7 Receipt of a Delay_Resp message from another PTP Instance

The logic for processing a Delay_Resp message shall be as defined in Figure 40. The states indicated in this figure refer to the current state of the PTP Port receiving the Delay_Resp message.

If the PTP Port receiving a Delay_Resp message is in the INITIALIZING or DISABLED states, the message shall be discarded. If the PTP Port receiving a Delay_Resp message is in the FAULTY state, the message shall be discarded except for implementation-specific purposes that otherwise meet the requirements of 9.2.

If the members of the requestingSourcePortIdentity field of the received Delay_Resp message are identical to the corresponding members of the sourcePortIdentity field of a prior Delay_Req message issued by the receiving PTP Instance, and the requestingSequenceId field of the received Delay_Resp message matches the sequenceId field of the same prior Delay_Req message, then the Delay_Resp message and the Delay_Req message are associated.

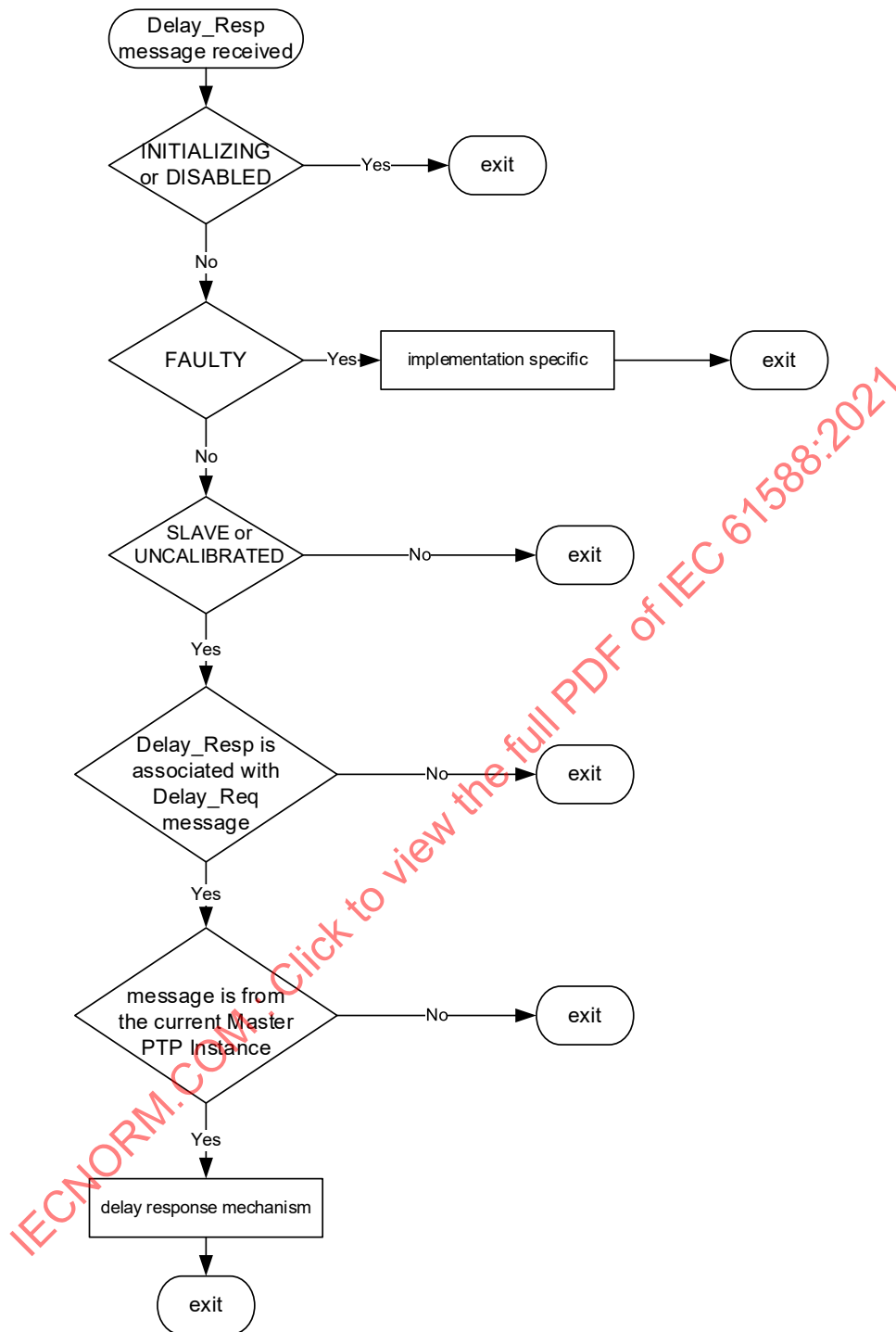


Figure 40—Receipt of Delay_Resp message logic

If the members of the sourcePortIdentity field of the received Delay_Resp message are identical to the corresponding members of the parentDS.parentPortIdentity member of the data set of the receiving PTP Instance, then the Delay_Resp message is from the current Parent PTP Instance.

Delay_Resp messages should be processed as soon as possible after receipt.

When the PTP Port receives a Delay_Resp message subsequent to transmitting the associated Delay_Req message, and all of the following conditions are met:

- The PTP Port receiving the Delay_Resp message is in the SLAVE or UNCALIBRATED state.
- The Delay_Resp message was received from the current Parent PTP Instance.
- The Delay_Resp message is associated with the transmitted Delay_Req message .

then the receiving PTP Instance shall:

- a) Execute the delay request-response mechanism actions required by 11.3, based on the contents of the received Delay_Resp message and associated Delay_Req message. The calculation of <meanPathDelay> may be based on a sequence of PTP timing messages. The calculation need not be performed on the receipt of every PTP timing message.
- b) Update the value of portDS.logMinDelayReqInterval member of the data set to the value of the logMessageInterval member of the Delay_Resp message.

9.5.8 Transmission of an Announce message

Unless otherwise stated in this standard, a PTP Port shall not transmit an Announce message except as required by this subclause.

A PTP Port in the MASTER state shall periodically transmit an Announce message.

Such Announce messages shall be transmitted as a multicast (see 7.3.1), such that the value of the arithmetic mean of the intervals, in seconds, between message transmissions is within $\pm 30\%$ of the value of $2^{\text{portDS.logAnnounceInterval}}$ (with portDS.logAnnounceInterval belonging to the data set of the transmitting PTP Port).

In addition, a PTP Port shall transmit Announce messages such that at least 90% of the inter-message intervals are within $\pm 30\%$ of the value of $2^{\text{portDS.logAnnounceInterval}}$. The interval between successive Announce messages should not exceed twice the value of $2^{\text{portDS.logAnnounceInterval}}$, to prevent causing an announceReceiptTimeout event.

NOTE 1—A minimum number of inter-message intervals is necessary to verify that a PTP Port meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detail discussion of statistical analyses, see Mood et al. [B40].

An optional unicast transmission at negotiated transmission intervals may be used for Announce messages subject to the terms of 16.1.

NOTE 2—The capacity of the PTP Port to maintain the granted announce inter-message period data for each unicast address can limit the number of unicast contracts that are permitted to be negotiated under 16.1. In some environments, multicast transmissions are not feasible. A unicast transmission is permitted provided the operation of the protocol is preserved (see 7.3.1).

9.5.9 Transmission of a Sync message

9.5.9.1 General specification

Unless otherwise stated in this standard, a PTP Port shall not transmit a Sync message except as required by 9.5.9.2 to 9.5.9.5.

9.5.9.2 General requirements

A PTP Port in the MASTER state shall periodically transmit a Sync message. Such Sync messages shall be transmitted as multicast (see 7.3.1), such that the value of the arithmetic mean of the intervals, in seconds, between PTP message transmissions is within $\pm 30\%$ of the value of $2^{\text{portDS.logSyncInterval}}$ (with $\text{portDS.logSyncInterval}$ belonging to the data set of the transmitting PTP Instance).

In addition, a PTP Instance shall transmit Sync messages such that at least 90% of the inter-message intervals are within $\pm 30\%$ of the value of $2^{\text{portDS.logSyncInterval}}$. The interval between successive PTP messages should not exceed twice the value of $2^{\text{portDS.logSyncInterval}}$.

NOTE 1—A minimum number of inter-message intervals is necessary to verify that a PTP Instance meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detail discussion of statistical analyses see Mood et al. [B40].

An optional unicast transmission at negotiated transmission intervals may be used for Sync messages subject to the terms of 16.1.

NOTE 2—The capacity of the PTP Node to maintain the granted Sync inter-message period data for each unicast address can limit the number of unicast contracts that are permitted to be negotiated under 16.1. In some environments, multicast transmissions are not feasible. A unicast transmission is permitted provided the operation of the protocol is preserved (see 7.3.1).

The fields of the transmitted Sync message shall conform to the requirements of Clause 13.

The `<syncEventEgressTimestamp>` meeting the requirements of 7.3.4 shall be generated upon transmission of the Sync message.

9.5.9.3 Additional requirements when the Message Length Extension option is active

If the applicable PTP Profile specifies that the Message Length Extension option (see 16.13) is active, then either:

- The PAD TLV defined in 14.4, or
- If defined by the applicable PTP Profile, a TLV listed in Table 52, for example, an `ORGANIZATION_EXTENSION_DO_NOT_PROPAGATE` TLV,

shall be appended as defined in 16.13 to ensure the `messageLength` requirements of 16.13 are met for egress PTP messages sent by a PTP port.

9.5.9.4 One-step PTP Ports

For a one-step egress PTP Port:

- The `originTimestamp` field of the Sync message shall be an estimate no worse than ± 1 s of the `<syncEventEgressTimestamp>` excluding any fractional nanoseconds, unless otherwise specified by the applicable PTP Profile.
- The `originTimestamp` field of the Sync message should be the `<syncEventEgressTimestamp>` excluding any fractional nanoseconds. The sum of the Sync message `correctionField` and `originTimestamp` field shall be the value of the `<syncEventEgressTimestamp>` including any fractional nanoseconds.

9.5.9.5 Two-step PTP Ports

For a two-step egress PTP Port:

- The `originTimestamp` field of the Sync message shall be 0 or an estimate no worse than ± 1 s of the `<syncEventEgressTimestamp>`, unless otherwise specified by the applicable PTP Profile.
- The `correctionField` of the Sync message shall be set to 0.
- The PTP Port shall transmit both a Sync and a Follow_Up message.
- The PTP Port shall capture the `sequenceId` value of the Sync message as an input to the `sequenceId` field of the Follow_Up message. The mechanism for obtaining the value for the `preciseOriginTimestamp` and `correctionField` fields of the associated Follow_Up message shall be started.

9.5.10 Transmission of a Follow_Up message

Unless otherwise stated in this standard, a PTP Port shall issue a Follow_Up message only if required by 9.5.9.5. The Sync message whose transmission requires the transmission of a Follow_Up message is the associated Sync message.

The Follow_Up message should be transmitted as soon as possible after the transmission of the associated Sync message and shall be transmitted prior to the transmission of a subsequent Sync message to the same destination address.

The value of the `sequenceId` field of the Follow_Up message shall be the value of the `sequenceId` field of the associated Sync message.

The value of the `preciseOriginTimestamp` field of the Follow_Up message shall be an estimate no worse than ± 1 s of the `<syncEventEgressTimestamp>` of the associated Sync message excluding any fractional nanoseconds, unless otherwise specified by the applicable PTP Profile.

The value of the `preciseOriginTimestamp` field of the Follow_Up message should be the `<syncEventEgressTimestamp>` of the associated Sync message excluding any fractional nanoseconds. The sum of the `correctionField` in the Follow_Up and associated Sync messages added to the `preciseOriginTimestamp` field of the Follow_Up message shall be the precise value of the `<syncEventEgressTimestamp>` of the associated Sync message including any fractional nanoseconds.

If the Follow_Up message is associated with an optional unicast Sync message per 9.5.9.2, then the Follow_Up message shall also be transmitted as a unicast PTP message to the same unicast address as the associated Sync message. These unicast Follow_Up messages shall meet all other requirements of this subclause.

9.5.11 Transmission of a Delay_Req message

9.5.11.1 General requirements

A PTP Instance shall issue a Delay_Req message on a PTP Port only if all of the following conditions are met:

- The PTP Port is in the SLAVE or UNCALIBRATED state.
- The PTP Instance is configured to execute the delay request-response mechanism (see 8.2.15.4.4).
- The PTP Instance is permitted to do so according to the timing requirements of 9.5.11.2.

Delay_Req messages shall be transmitted as multicast except if:

- a) The optional unicast provisions of 16.1 and/or 16.9 are used
- b) Specified otherwise by the applicable PTP Profile

An optional unicast transmission at negotiated transmission intervals may be used for Delay_Req messages subject to the terms of 16.1.

The fields of the Delay_Req message shall conform to the requirements of 11.3.

NOTE—If the PTP Instance uses a PTP Profile that specifies synthesize only, then the PTP Instance is not required to send Delay_Req messages. In this case, maintaining delay request state information is not required.

9.5.11.2 Timing requirements

Unless the unicast option of 16.1 is in use, the transmission of Delay_Req messages from a PTP Port shall be limited as follows:

- a) The initial Delay_Req message may be transmitted when required.
- b) Subsequent Delay_Req messages shall be transmitted such that the value of the arithmetic mean of the intervals, in seconds between Delay_Req message transmissions is not less than the value of $0.9 \times 2^{\text{portDS.logMinDelayReqInterval}}$ in seconds.

NOTE 1—The portDS.logMinDelayReqInterval value is the logMessageInterval field of the last Delay_Resp message received in response to a Delay_Req message issued by the PTP Port.

NOTE 2—A minimum number of inter-message intervals is necessary to verify that a PTP Instance meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detail discussion of statistical analyses, see Mood et al. [B40].

- c) While meeting the mean interval between Delay_Req message transmission specification, the transmission times for Delay_Req messages shall use one of the following timing options:
 - 1) Transmission times shall be selected such that the interval between successive Delay_Req messages is taken from a random distribution. Unless otherwise specified in the applicable PTP Profile, the random distribution shall be a uniform random distribution with a minimum value of 0 and a maximum value of $\{2^{\text{logMinDelayReqInterval}+1}\}$ seconds. A new random value for the transmission interval shall be computed for each PTP message transmitted. The granularity of this distribution is implementation-specific but shall be no greater than $2^{\text{logSyncInterval}-4}$ s. This option should be used when using the multicast communication model, or the mixed multicast/unicast communication model, and may be used with a unicast model (see 7.3.1). For example, for a value of $\text{logMinDelayReqInterval} = \text{logSyncInterval}$ (see 7.7.2.4), the distribution has a minimum value of 0 and a maximum value of $\{2^{\text{logSyncInterval}+1}\} = 2 \text{ syncInterval}$; that is, the distribution has a maximum value of 2 syncInterval, and the resulting mean transmission interval is 1 syncInterval.
 - 2) Delay_Req messages should be transmitted as soon as possible following the receipt of a Sync message. This option may be used with a unicast model (see 7.3.1). It shall not be used when using the multicast model unless specified in the applicable PTP Profile.

NOTE 3—If the timestamps at the PTP Ports in the MASTER and SLAVE states are taken relative to clocks whose frequencies differ from the frequency of the Grandmaster Clock, the mean propagation delay computed using the delay request-response mechanism will be in error. This error has two components: (a) a component due to any difference between the frequencies of the Master Clock and the Slave Clock, and (b) a component due to the difference between either the frequency of the Master Clock or the Slave Clock from the frequency of the Grandmaster Clock. Component (a) is equal to one-half the fractional frequency offset (see 3.1.15) between the frequencies at the Master Clock and Slave Clock used to take the respective timestamps, multiplied by the time interval between the receipt of the most

recent Sync message by the Slave PTP Instance and its sending of a Delay_Req (this time interval is the turnaround time). The purpose of the above timing option of transmitting a Delay_Req message as soon as possible after receiving a Sync message is to reduce the magnitude of error component (a). However, if this option is used when a multicast transmission model is used, there exists the possibility that many PTP Ports in the SLAVE state will send a Delay_Req message on receipt of a multicast Sync message and overwhelm the Master PTP Instance with multiple Delay_Req messages. This is mainly a concern if the multicast model is used in a PTP Network that contains Transparent Clocks, devices that do not support PTP (e.g., non-PTP bridges or routers), and/or media where the PTP messages are not sent point-to-point; it is not a concern in PTP Networks that consist only of Boundary Clocks and Ordinary Clocks and where the PTP transmission is point-to-point. For these reasons, this timing option is allowed for unicast networks, and for multicast networks if specified in the applicable PTP Profile. Note that a similar error in mean propagation delay can occur if the peer-to-peer delay mechanism is used (see 6.6.4). Note also that this error is separate from any error due to delay asymmetry.

An optional unicast transmission at negotiated transmission intervals may be used for Delay_Req and Delay_Resp messages subject to the terms of 16.1.

NOTE 4—The capacity of the PTP Node to maintain the granted Delay_Resp inter-message period data for each unicast address can limit the number of unicast contracts that are permitted to be negotiated under 16.1. In some environments, multicast transmissions might not be feasible. A unicast transmission is permitted provided the operation of the protocol is preserved (see 7.3.1).

9.5.12 Transmission of a Delay_Resp message

Unless otherwise stated in this standard, a PTP Instance shall issue a Delay_Resp message on a PTP Port when all of the following conditions are met:

- a) The PTP Port is in the MASTER state or is required as the result of an optional unicast contract (see 16.1), and
- b) The device is configured to execute the delay request-response mechanism (see 8.2.15.4.4), and
- c) The transmission is the result of the receipt of the associated Delay_Req message (see 9.5.6), and
- d) When the transmission of Delay_Req messages is in accordance with the requirements of 9.5.11.2.

NOTE 1—With respect to requirement a), the provisions of the optional 17.3 (Alternate master) allow transmission of a Delay_Resp message from a PTP Port that is not in the MASTER state.

NOTE 2—Requirement d) is applicable when the capacity of the PTP Instance receiving the Delay_Req messages (e.g., number of PTP Ports in the SLAVE state simultaneously supported) is not exceeded.

NOTE 3—There might be short periods when the arithmetic mean of the intervals between the reception of successive Delay_Req messages is less than the value of $0.9 \times 2^{\text{portDS.logMinDelayReqInterval}}$ in seconds specified in 9.5.11.2; the PTP Instance is expected to tolerate these periods and respond with Delay_Resp messages.

- e) The Delay_Req message whose reception may require the transmission of a Delay_Resp message is the associated Delay_Req message.

Delay_Resp messages shall be transmitted as multicast if the associated Delay_Req message was sent as multicast.

Delay_Resp messages shall be transmitted as unicast if the associated Delay_Req message was sent as unicast, subject to the provisions of 16.9.

The fields of the Delay_Resp message shall conform to the requirements of 11.3.

The receiveTimestamp field of the Delay_Resp shall be the <delayReqEventIngressTimestamp> of the associated Delay_Req message.

Prior to transmitting the Delay_Resp message, the PTP Port shall, in order:

- Compute the updated value for the logMinDelayReqInterval field (see 7.7.2.4).
- Use this computed value to update the value of portDS.logMinDelayReqInterval member of the data set of the PTP Port transmitting the Delay_Resp message (see 8.2.15.3.2).
- Ensure the PTP message field values are the current values as specified in 11.3.2.
- Insert the value as specified in Table 42 into logMessageInterval field of the message.

The Delay_Resp message should be transmitted as soon as possible after the receipt of the associated Delay_Req message.

9.5.13 Transmission of a Pdelay_Req message

9.5.13.1 General requirements

A PTP Instance shall issue a Pdelay_Req message only if all of the following conditions are met:

- The PTP Port is configured to execute the peer-to-peer delay mechanism (see 8.2.15.4.4).
- The PTP Port transmission of the Pdelay_Req message meets the timing requirements of 9.5.13.2.

The fields of this Pdelay_Req message shall conform to the requirements of 11.4.2.

NOTE—If the PTP Instance uses a PTP Profile that specifies synthesize only, then the PTP Instance is not required to invoke either the peer-to-peer delay or delay request-response mechanism. In this case, maintaining peer-to-peer delay mechanism state information is not required.

9.5.13.2 Timing requirements

The transmission of Pdelay_Req messages from a requesting PTP Port shall be limited as follows:

- The initial Pdelay_Req message may be transmitted when required.
- Subsequent Pdelay_Req messages shall be transmitted such that the value of the arithmetic mean of the intervals, in seconds, between Pdelay_Req message transmissions is not less than the value of $0.9 \times 2^{\text{portDS.logMinPdelayReqInterval}}$.

NOTE—A minimum number of inter-message intervals is necessary to verify that a PTP Instance meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For a more detailed discussion of statistical analyses, see Mood et al. [B40].

9.5.14 Transmission of Pdelay_Resp message

A Pdelay_Resp message should be transmitted as soon as possible after the receipt of the associated Pdelay_Req message.

9.5.15 Transmission of Pdelay_Resp_Follow_Up message

A Pdelay_Resp_Follow_Up message should be transmitted as soon as possible after the transmission of the associated Pdelay_Resp message.

9.6 Changes in the PTP Instance

Changes due to 1) internal properties of the Local PTP Clock, for example, a disruption of a local oscillator; or 2) interactions outside of PTP, for example, changes in information received from a GPS receiver, to which the Local PTP Clock is directly synchronized, may result in updates to the data sets.

All resulting changes to data sets shall be treated atomically with respect to any other activity accessing the data sets, including the activity specified in 9.2.6.9, except when the specifications of 17.6 are enabled; in which case, the activity specified in 9.2.6.9 is not in effect (see 17.6.5.3).

Unless otherwise stated in the standard or the applicable PTP Profile, PTP messages transmitted by different PTP Ports of a PTP Instance may reflect the updated dataset values at different times.

10. PTP for Transparent Clocks

10.1 Requirements for both end-to-end and peer-to-peer Transparent Clocks

10.1.1 Residence time computations

The residence time for each PTP event message shall be computed for each egress PTP Port as follows:

$$\langle \text{residenceTime} \rangle = \langle \text{egress timestamp} \rangle - \langle \text{ingress timestamp} \rangle$$

where $\langle \text{egress timestamp} \rangle$ and $\langle \text{ingress timestamp} \rangle$ are the egress and ingress timestamps pertinent to the specific PTP event message, for example, for a Sync message $\langle \text{syncEventEgressTimestamp} \rangle$ and $\langle \text{syncEventIngressTimestamp} \rangle$ (see 7.3.4 for further details on timestamp generation).

10.1.2 Retransmission of PTP messages that contain TLVs

A TLV attached to a PTP message is considered part of the PTP message (see 13.2 and 13.4) for the purposes of retransmission of the PTP message.

For the purposes of retransmission of the PTP message, if a Transparent Clock implements and enables an optional feature requiring the use of a TLV, the Transparent Clock can remove the TLV if required by the specification of such optional feature.

10.2 End-to-end Transparent Clock requirements

10.2.1 General requirements

All PTP messages shall be retransmitted in conformance with 7.3.1.

Except as noted in the following subclauses of 10.2, no changes in the PTP message common headers shall be made in the process of retransmission.

An end-to-end Transparent Clock shall not implement the peer-to-peer delay mechanism of 11.4.

An end-to-end Transparent Clock shall not correct for path delay on either ingress or egress PTP Ports.

The implementation shall process the following PTP messages as associated pairs:

- Sync and Follow_Up
- Delay_Req and Delay_Resp
- Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up

The above pairs of PTP messages shall constitute associated pairs if and only if they meet the following constraints:

- a) For all such message pairs, their domainNumber fields match and their sdoId fields match. In the case where a PTP message is transmitted per IEEE Std. 1588-2008, the transport specific field and a field with all bits 0 substitutes for the sdoId field in determining the match, see Table 2.
- b) For all such message pairs, their sequenceId fields match.
- c) For the message pair Sync and Follow_Up, their sourcePortIdentity fields match.
- d) For the message pair Delay_Req and Delay_Resp, the requestingPortIdentity field of the Delay_Resp message matches the sourcePortIdentity field of the Delay_Req message.
- e) For the message pairs Pdelay_Req Pdelay_Resp and Pdelay_Resp_Follow_Up, the requestingPortIdentity of Pdelay_Resp and, if applicable, the Pdelay_Resp_Follow_Up messages, matches the sourcePortIdentity field of the Pdelay_Req message.

NOTE 1—There are cases where domainNumber and sdoId need not be tested because the received PTP message can be treated as a singleton and not as a member of a pair. For example, if the Transparent Clock can make residence time corrections to Sync messages, then the correction for residence time can be made on the correctionField of the Sync message, irrespective of whether there is or is not a Follow_Up message, and therefore there is no need to check the association. This sort of example might not apply in the presence of additional requirements, (e.g., security) imposed in a profile or by the standard.

NOTE 2—Even though subclause 10.2 deals with End-to-End Transparent Clocks, the specifications for handling Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up messages are needed to implement the specifications of subclause 10.2.3.3 and subclause 10.2.3.

There are two PTP message processing options for end-to-end Transparent Clocks. End-to-end Transparent Clocks shall meet the specifications of one of the following two options:

- f) The specifications of 10.2.2.
- g) The specifications of 10.2.3.

NOTE 3—Option “g)” is intended for architectures, for example, blade architectures, and implementations that require more flexibility in where the corrections for <residenceTime> and <delayAsymmetry> are made. For example, option “g)” allows these corrections to be made by the ingress or egress PTP Ports or by the PTP Common Core of the PTP Instance. Likewise, the corrections can be made using any applicable correctionField or timestamp field provided the constraints specified in option “g)” are met. End-to-end Transparent Clocks meeting the specifications of option “f)” automatically meet the specifications of option “g)”.

10.2.2 Default PTP message processing in end-to-end Transparent Clocks

Except as noted in the following subclauses, no changes in the PTP message common headers shall be made in the process of retransmission.

10.2.2.1 Processing of Sync messages

10.2.2.1.1 Egress one-step PTP Ports

The <residenceTime> (see 10.1.1), and the ingress path <delayAsymmetry> (see 7.4.2), shall be added to the correctionField of the Sync message by the egress PTP Port of the Transparent Clock prior to retransmission of the Sync message on the egress PTP Port. The egress PTP Port shall make any needed corrections to checksums or other content dependent fields of the PTP message.

NOTE—The data type of the correctionField allows the <residenceTime> and <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

No modification of any received Follow_Up messages shall be made.

10.2.2.1.2 Egress two-step PTP Ports

10.2.2.1.2.1 Ingress Sync message twoStep flag is FALSE

If the twoStepFlag of the Sync message received from the ingress PTP Port of the Transparent Clock is FALSE indicating that no Follow_Up message will be received, then:

- a) The twoStepFlag of the retransmitted Sync message shall be set to TRUE to indicate that a Follow_Up message will follow. The Sync message should be retransmitted on the egress PTP Port as soon as possible, with any needed corrections to checksums or other content-dependent fields of the PTP message. This modified Sync message shall be used to generate the egress timestamp for the computation of <residenceTime> for the Sync message.
- b) A Follow_Up message shall be prepared for transmission on the egress PTP Port as follows:
 - 1) The originTimestamp of the received Sync message shall be copied into the preciseOriginTimestamp field of the Follow_Up message.
 - 2) The common header of the Follow_Up message shall be identical to the common header of the received Sync message except for
 - i) The PTP messageType, which shall be set per 13.3.2.3.
 - ii) The correctionField, which shall be set to the <residenceTime> plus the value of the ingress path <delayAsymmetry> (see 7.4.2), associated with the received Sync message.

NOTE—The data type of the correctionField allows the <residenceTime> and the <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

10.2.2.1.2.2 Ingress Sync message twoStep flag is TRUE

If the twoStepFlag of the Sync message received from the ingress PTP Port of the Transparent Clock is TRUE, indicating that a Follow_Up message will be received, then:

- a) The received Sync message shall be retransmitted from the egress PTP Port. This Sync message shall be used to generate the egress timestamp for the computation of <residenceTime> for the Sync message.
- b) The <residenceTime> plus the value of the ingress path <delayAsymmetry> (see 7.4.2), associated with the received Sync message, shall be added to the correctionField of the Follow_Up message associated with the Sync message prior to retransmission of the Follow_Up message on the egress

PTP Port. With the exception of the correctionField, the common header of the Follow_Up message shall be identical to the common header of the received Follow_Up message. The egress PTP Port shall make any needed corrections to checksums or other content-dependent fields of the PTP message.

NOTE—To correctly associate Sync and Follow_Up messages requires that the Transparent Clock maintain a record of the sourcePortIdentity and sequenceId fields of the Sync message for comparison with the sourcePortIdentity and sequenceId fields of Follow_Up messages. The data type of the correctionField allows the <residenceTime> and the <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

10.2.2.2 Processing of Delay_Req messages

10.2.2.2.1 General specification

Subclause 10.2.2 applies to end-to-end Transparent Clocks. Peer-to-peer Transparent Clocks do not support Delay_Req messages.

Delay_Req messages shall not be corrected for asymmetry for the path connected to an ingress PTP Port.

10.2.2.2.2 Egress one-step PTP Ports

The egress PTP Port shall retransmit the Delay_Req message and shall generate the egress timestamp for the computation of the <residenceTime> for the Delay_Req message.

The <residenceTime> (see 10.1.1) of the Delay_Req message, minus the value of the egress path <delayAsymmetry> (see 7.4.2), shall be added to the correctionField of the Delay_Req message by the egress PTP Port of the Transparent Clock prior to the retransmission of the Delay_Req message. The egress PTP Port shall make any needed corrections to checksums or other content-dependent fields of the PTP message.

NOTE—The data type of the correctionField allows the <residenceTime> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

No modification of any received Delay_Resp messages shall be made.

10.2.2.2.3 Egress two-step PTP Ports

The received Delay_Req message shall be retransmitted from the egress PTP Port. This Delay_Req message shall be used to generate the egress timestamp for the computation of <residenceTime> for the Delay_Req message.

The <residenceTime> (see 10.1.1), minus the value of the egress path <delayAsymmetry> (see 7.4.2), shall be added to the correctionField of the Delay_Resp message associated with the Delay_Req message prior to transmission of the Delay_Resp message on the egress PTP Port, which is the ingress PTP Port for the Delay_Req message. The egress PTP Port of the Delay_Resp message shall make any needed corrections to checksums or other content dependent fields of the Delay_Resp message.

NOTE—To correctly associate Delay_Req and Delay_Resp messages requires that the Transparent Clock maintain a record of the sourcePortIdentity and sequenceId fields of the Delay_Req message for comparison with the requestingPortIdentity and sequenceId fields of Delay_Resp messages. The data type of the correctionField allows the <residenceTime> and <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

10.2.2.3 Processing of Pdelay_Req and Pdelay_Resp messages

10.2.2.3.1 General

Subclause 10.2.2.3 applies only to end-to-end Transparent Clocks. Pdelay_Req and Pdelay_Resp messages terminate at peer-to-peer Transparent Clocks.

Ingress PTP Ports receiving a Pdelay_Req or Pdelay_Resp message shall generate the ingress timestamps used to compute the <residenceTime> for these PTP messages.

Egress PTP Ports retransmitting a Pdelay_Req or Pdelay_Resp message shall generate the egress timestamps used to compute the <residenceTime> for these PTP messages.

NOTE 1—Peer-to-peer Transparent Clocks are normally used only in a homogeneous PTP Network of PTP Instances implementing peer-to-peer semantics. The provisions of 10.2.2.3 allow the use of end-to-end Transparent Clocks in PTP Networks based on future versions of the standard that might specify how to implement mixed systems with one-to-many connections between PTP Instances.

NOTE 2—In the 2008 edition, corrections for asymmetry were applied to PTP messages associated with the peer-to-peer mechanism. Analysis shows that the net result of these corrections was zero. Therefore, in this edition, asymmetry corrections for the peer-to-peer PTP messages in end-to-end Transparent Clocks have been omitted.

10.2.2.3.2 Egress one-step PTP Ports on end-to-end Transparent Clocks

The <residenceTime> of the Pdelay_Req message shall be added to the correctionField of the Pdelay_Req message by the egress PTP Port of the Transparent Clock as the Pdelay_Req message is being retransmitted. The egress PTP Port shall make any needed corrections to checksums or other content-dependent fields of the PTP message.

No modification of any received Pdelay_Resp_Follow_Up or Pdelay_Resp messages shall be made for the <residenceTime> of a Pdelay_Req message. For the Pdelay_Resp message, the value of the twoStepFlag shall remain unchanged.

NOTE—This requirement on the twoStepFlag removes the inconsistency with respect to Table 20 of the previous edition.

The <residenceTime> of a Pdelay_Resp message shall be added to the correctionField of the Pdelay_Resp message by the egress PTP Port of the Transparent Clock as the Pdelay_Resp message is being retransmitted. The egress PTP Port shall make any needed corrections to checksums or other content-dependent fields of the PTP message.

No modification of any received Pdelay_Resp_Follow_Up messages shall be made for the <residenceTime> of a Pdelay_Resp message.

10.2.2.3.3 Egress two-step PTP Ports on end-to-end Transparent Clocks

The <residenceTime> of the Pdelay_Req message shall be measured and saved for incorporation into the correctionField of a Pdelay_Resp_Follow_Up message associated with the Pdelay_Req message.

The <residenceTime> of a Pdelay_Resp message associated with the Pdelay_Req message shall be measured and saved for incorporation into the correctionField of the Pdelay_Resp_Follow_Up message associated with the Pdelay_Req message.

If the twoStepFlag of the Pdelay_Resp message associated with the Pdelay_Req message received on the ingress PTP Port is FALSE, indicating that no Pdelay_Resp_Follow_Up message will be received, then:

- a) The Pdelay_Resp message should be retransmitted as soon as possible, with any needed corrections to checksums or other content-dependent fields of the PTP message. The twoStepFlag of the retransmitted Pdelay_Resp message shall be set to TRUE to indicate that a Pdelay_Resp_Follow_Up message will follow. This modified Pdelay_Resp message shall be used to generate the egress timestamp for the computation of the <residenceTime> for the Pdelay_Resp message.
- b) A Pdelay_Resp_Follow_Up message shall be prepared for transmission as follows:
 - 1) The common header of the Pdelay_Resp_Follow_Up message shall be identical to the common header of the received Pdelay_Resp message except for:
 - i) The PTP messageType, which shall be set per 13.3.2.3.
 - ii) The correctionField of the Pdelay_Resp_Follow_Up message, which shall be set to the sum of the <residenceTime> of the Pdelay_Resp and the <residenceTime> of the Pdelay_Req message associated with the Pdelay_Resp message.
 - 2) Copy the requestingPortIdentity field from the Pdelay_Resp message received on the ingress PTP Port to the requestingPortIdentity field of the Pdelay_Resp_Follow_Up message.
 - 3) Set the responseOriginTimestamp to 0.

If the twoStepFlag of the Pdelay_Resp message received on the ingress PTP Port is TRUE, indicating that a Pdelay_Resp_Follow_Up message will be received, then:

- c) The received Pdelay_Resp message shall be retransmitted from the egress PTP Port. This Pdelay_Resp message shall be used to generate the egress timestamp for the computation of residence time for the Pdelay_Resp message. The twoStepFlag of the retransmitted Pdelay_Resp message shall remain set to TRUE to indicate that a Pdelay_Resp_Follow_Up message will follow.
- d) The sum of the <residenceTime> of the Pdelay_Resp and the <residenceTime> of the Pdelay_Req message associated with the Pdelay_Resp message shall be added to the correctionField of the Pdelay_Resp_Follow_Up message associated with the Pdelay_Req message prior to transmission on the egress PTP Port.

NOTE 1—To correctly associate Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages requires that the Transparent Clock maintain a record of the sourcePortIdentity, and sequenceId fields of the Pdelay_Req message for comparison with the requestingPortIdentity and sequenceId fields of Pdelay_Resp and Pdelay_Resp_Follow_Up messages.

NOTE 2—The data type of the correctionField allows the <residenceTime> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

10.2.3 Alternate processing of Sync, Delay_Req, Delay_Resp, Pdelay_Req, and Pdelay_Resp messages on end-to-end Transparent Clocks

10.2.3.1 General requirements

When processing Sync, Delay_Req, Delay_Resp, Pdelay_Req, and Pdelay_Resp messages, corrections for <delayAsymmetry> and <residenceTime> may be implemented in the ingress PTP Port, egress PTP Ports, the PTP Common Core, or any combination and using any of the available timestamp and correction fields subject to the restrictions of 10.2.3.2, 10.2.3.3 and 10.2.3.4.

NOTE—The above provides a context for implementations to have more flexibility in meeting the terms of 10.2.3.2, 10.2.3.3 and 10.2.3.4. Egress PTP Ports need to have access to the following information to fulfill these terms: a) which corrections have been done in the ingress PTP Port or the PTP Common Core or the information to make the corrections in the egress PTP Port, b) the value of the applicable ingress timestamp or a specification on how to correct for <residenceTime> based on actions by the ingress PTP Port.

In all cases, the twoStepFlag of the Sync and Pdelay_Resp message received at egress PTP Port on the

Transparent Clock indicates whether or not for Sync and Pdelay_Resp messages that an associated Follow_Up or Pdelay_Resp_Follow_Up, respectively, will be received from the ingress PTP Port.

If the twoStepFlag of Sync or Pdelay_Resp messages received by the egress PTP Port is FALSE, then:

- a) If the egress PTP Port is a one-step PTP Port, then after implementing the corrections of 10.2.3.2 or 10.2.3.4, respectively, it shall retransmit the received Sync or Pdelay_Resp message.
- b) If the egress PTP Port is a two-step PTP Port, then after implementing the corrections of 10.2.3.2 or 10.2.3.4, respectively, it shall retransmit the received Sync or Pdelay_Resp message and generate and transmit an associated Follow_Up or Pdelay_Resp_Follow_Up message, respectively.

If the twoStepFlag of Sync or Pdelay_Resp message received by the egress PTP Port is TRUE, then:

- c) If the egress PTP Port is a one-step PTP Port, it shall wait for the associated Follow_Up or Pdelay_Resp_Follow_Up message, respectively, from the ingress PTP Port of the Transparent Clock and then after implementing the corrections of 10.2.3.2 or 10.2.3.4, respectively, retransmit the Sync or Pdelay_Resp message.
- d) If the egress PTP Port is a two-step PTP Port, it shall, after implementing the corrections of 10.2.3.2 or 10.2.3.4 respectively:
 - 1) Retransmit the Sync or Pdelay_Resp message,
 - 2) Wait for and then retransmit the associated Follow_Up or Pdelay_Resp_Follow_Up message received, respectively, from the ingress PTP Port of the Transparent Clock.

In all cases, the twoStepFlag for Sync and Pdelay_Resp messages transmitted from the egress PTP Port shall be TRUE if an associated Follow_Up or Pdelay_Resp_Follow_Up, respectively, will follow and FALSE otherwise. For all other PTP message types, the twoStepFlag of PTP messages transmitted from the egress PTP Port shall be FALSE.

In all cases, the ingress and egress PTP Ports may make use of the messageTypeSpecific field of the common header as specified in 13.3.2.10.

In all cases, egress PTP Ports shall make any needed corrections to checksums or other content-dependent fields of the PTP message leaving the egress PTP Port.

In 10.2.3.2, 10.2.3.3, and 10.2.3.4, the constraint notation <ingress/egress> [<Message type-1> (<fieldname>) + <Message type-2>] indicates:

- e) Whether the PTP messages involved are those received by an ingress PTP Port or transmitted or retransmitted by an egress PTP Port of the Transparent Clock.
- f) The name of the PTP message, for example, Sync, and the name of the field of the named PTP message, for example, originTimestamp.

10.2.3.2 Constraints on processing of Sync and Follow_Up messages

In the following constraint specifications, ingress twoStepFlag and egress twoStepFlag refer to the twoStepFlag of a Sync message received on an ingress PTP Port and the corresponding Sync message retransmitted on the egress PTP Port, respectively.

The <residenceTime> is that of the Sync message.

The following constraints shall be satisfied for the respective cases.

10.2.3.2.1 All cases

Except for the twoStepFlag and the correctionField, the common header of Sync messages shall not be changed from ingress to egress. Except for the twoStepFlag, which shall be FALSE, and the correctionField, the common header of an egress Follow_Up message, if present, shall be identical to the common header of the associated Sync message.

10.2.3.2.2 Ingress and egress twoStepFlags both FALSE

$$\text{Egress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] = \text{ingress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] + \text{ingress} [<\text{delayAsymmetry}>] + <\text{residenceTime}>$$

10.2.3.2.3 Ingress twoStepFlag FALSE and egress twoStepFlag TRUE

$$\text{Egress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correctionField})] = \text{ingress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] + \text{ingress} [<\text{delayAsymmetry}>] + <\text{residenceTime}>$$

10.2.3.2.4 Ingress twoStepFlag TRUE and egress twoStepFlag FALSE

$$\text{Egress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] = \text{ingress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correctionField})] + \text{ingress} [<\text{delayAsymmetry}>] + <\text{residenceTime}>$$

10.2.3.2.5 Ingress twoStepFlag TRUE and egress twoStepFlag TRUE:

$$\text{Egress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correctionField})] = \text{ingress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correctionField})] + \text{ingress} [<\text{delayAsymmetry}>] + <\text{residenceTime}>$$

10.2.3.3 Constraints on processing of Delay_Req and Delay_Resp messages

The <residenceTime> is that of the Delay_Req message. The egress[<delayAsymmetry>] is the asymmetry of the egress path of the Delay_Req message.

The following constraint shall be satisfied:

$$\text{Egress} [\text{Delay\_Resp}(\text{correctionField})] - \text{ingress} [\text{Delay\_Resp}(\text{correctionField})] + \text{egress} [\text{Delay\_Req}(\text{correctionField})] - \text{ingress} [\text{Delay\_Req}(\text{correctionField})] = <\text{residenceTime}> - \text{egress} [<\text{delayAsymmetry}>]$$

where the Delay_Resp message is associated with the Delay_Req message.

The common header with the exception of the correctionField shall not be changed.

10.2.3.4 Constraints on processing of Pdelay_Req and Pdelay_Resp messages

In the following constraint specifications, the ingress twoStepFlag refers to the twoStepFlag of the Pdelay_Resp message received on the ingress PTP Port of the Pdelay_Resp message that is associated with the Pdelay_Req message.

The following constraints shall be satisfied for the respective cases.

10.2.3.4.1 All cases

For Pdelay_Req and Pdelay_Resp messages, the common headers, with the exception of the correctionField, shall not be changed.

Except for the twoStepFlag, which shall be FALSE, the correctionField, and the messageType, which shall be set per 13.3.2.3, the common header of an egress Pdelay_Resp_Follow_Up message, if present, shall be identical to the common header of the associated Pdelay_Resp message.

10.2.3.4.2 Ingress twoStep flag is FALSE

$$\begin{aligned} & \text{Egress [Pdelay\_Resp(requestReceiptTimestamp)]} - \text{ingress} \\ & [\text{Pdelay\_Resp(requestReceiptTimestamp)}] + \text{egress [Pdelay\_Resp(correctionField)]} - \text{ingress} \\ & [\text{Pdelay\_Resp(correctionField)}] + \text{egress [Pdelay\_Req(correctionField)]} - \\ & \text{ingress[Pdelay\_Req(correctionField)]} = (<\text{residenceTime}> \text{ of Pdelay\_Resp message}) \\ & + (<\text{residenceTime}> \text{ of Pdelay\_Req message}) \end{aligned}$$

10.2.3.4.3 Ingress twoStep flag is TRUE

$$\begin{aligned} & \text{Egress[Pdelay\_Resp\_Follow\_Up(responseOriginTimestamp)]} \\ & - \text{ingress[Pdelay\_Resp\_Follow\_Up(responseOriginTimestamp)]} \\ & + \text{egress[Pdelay\_Resp\_Follow\_Up(correctionField)]} \\ & - \text{ingress[Pdelay\_Resp\_Follow\_Up(correctionField)]} + \text{egress} \\ & [\text{Pdelay\_Resp(requestReceiptTimestamp)}] - \text{ingress [Pdelay\_Resp(requestReceiptTimestamp)]} \\ & + \text{egress [Pdelay\_Resp(correctionField)]} - \text{ingress [Pdelay\_Resp(correctionField)]} + \text{egress} \\ & [\text{Pdelay\_Req(correctionField)}] - \text{ingress[Pdelay\_Req(correctionField)]} = (<\text{residenceTime}> \text{ of} \\ & \text{Pdelay\_Resp message}) + (<\text{residenceTime}> \text{ of Pdelay\_Req message}) \end{aligned}$$

10.3 Peer-to-peer Transparent Clock requirements

10.3.1 General requirements

All Announce, Sync, Follow_Up, PTP management, and Signaling messages shall be retransmitted in conformance with 7.3.1.

Except as noted in the following subclauses of 10.3, no changes in the PTP message common headers shall be made in the process of retransmission.

All PTP Delay_Req and Delay_Resp messages should be discarded.

The peer-to-peer delay mechanism must be implemented as specified in 11.4.

The implementation shall process Sync and Follow_Up messages in associated pairs. Sync and Follow_Up shall constitute an associated pair if and only if they meet the following constraints:

- a) Their domainNumber fields match and their sdoId fields match. In the case where a PTP messages is transmitted per IEEE Std 1588-2008, the transport specific field and a field with all bits 0 substitutes for the sdoId field in determining the match; see Table 2.
- b) Their sequenceId fields match and their sourcePortIdentity fields match.

NOTE 1—There are cases where domainNumber and sdoId need not be tested because the received PTP message can be treated as a singleton and not as a member of a pair. For example, if the Transparent Clock can make residence time corrections to Sync messages, then the correction for residence time can be made on the correctionField of the Sync message, irrespective of whether there is or is not a Follow_Up message, and therefore there is no need to check the association. This sort of example might not apply in the presence of additional requirements, (e.g., security) imposed in a profile or by the standard.

There are two PTP message processing options for Sync and, if required, Follow_Up messages on peer-to-peer Transparent Clocks. Peer-to-peer Transparent Clocks shall meet the specifications of one of the following two options when handling Sync and Follow_Up messages:

- c) The specifications of 10.3.2,
- d) The specifications of 10.3.3.

NOTE 3—Option d) is intended for architectures, for example, blade architectures, and implementations that require more flexibility in where the corrections for <residenceTime>, <delayAsymmetry>, and <meanLinkDelay> are made. For example, option d) allows these corrections to be made by the ingress or egress PTP Ports or by the PTP Common Core of the Transparent Clock. Likewise, the corrections can be made using any applicable correctionField or timestamp field provided the constraints specified in option d) are met. Peer-to-peer Transparent Clocks meeting the specifications of option c) automatically meet the specifications of option d).

10.3.2 Default Sync and Follow_Up message processing in peer-to-peer Transparent Clocks

10.3.2.1 Egress one-step PTP Ports

The values of the <residenceTime> (see 10.1.1), the ingress path <delayAsymmetry> (see 7.4.2), and the <meanLinkDelay>, as measured by peer-to-peer delay mechanism for the PTP Link connected to the ingress PTP Port on which the Sync message was received (see 7.4.2), shall be added to the correctionField of the Sync message by the egress PTP Port of the PTP Instance prior to retransmission of the Sync

message on the egress PTP Port. The egress PTP Port shall make any needed corrections to checksums or other content-dependent fields of the PTP message.

NOTE—The data type of the correctionField allows the <residenceTime> and the <meanLinkDelay> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

No modification of any received Follow_Up messages shall be made.

10.3.2.2 Egress two-step PTP Ports

10.3.2.2.1 Ingress Sync message twoStep flag is FALSE

If the twoStepFlag of the Sync message received from the ingress PTP Port of the Transparent Clock is FALSE indicating that no Follow_Up message will be received, then:

- a) The twoStepFlag of the retransmitted Sync message shall be set to TRUE to indicate that a Follow_Up message will follow. The Sync message should be retransmitted on the egress PTP Port as soon as possible, with any needed corrections to checksums or other content-dependent fields of the PTP message. This modified Sync message shall be used to generate the egress timestamp for the computation of <residenceTime> for the Sync message.
- b) A Follow_Up message shall be prepared for transmission on the egress PTP Port as follows:
 - 1) The originTimestamp of the received Sync message shall be copied into the preciseOriginTimestamp field of the Follow_Up message.
 - 2) The common header of the Follow_Up message shall be identical to the common header of the received Sync message except for:
 - i) The messageType, which shall be set per 13.3.2.3.
 - ii) The correctionField, which shall be set to the <residenceTime> (see 10.1.1), plus the sum of the values of the <meanLinkDelay> as measured by peer-to-peer delay mechanism and the ingress path <delayAsymmetry> (see 7.4.2), for the PTP Link connected to the ingress PTP Port on which the Sync message was received (see 7.4.2).

NOTE—The data type of the correctionField allows the <residenceTime>, <meanLinkDelay>, and the <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

10.3.2.2.2 Ingress Sync message twoStep flag is TRUE

If the twoStepFlag of the Sync message received from the ingress PTP Port of the Transparent Clock is TRUE, indicating that a Follow_Up message will be received, then:

- a) The received Sync message shall be retransmitted from the egress PTP Port. This Sync message shall be used to generate the egress timestamp for the computation of <residenceTime> for the Sync message.
- b) The <residenceTime> (see 10.1.1), plus the values of the ingress path <meanLinkDelay> and the <delayAsymmetry> (see 7.4.2), associated with the received Sync message shall be added to the correctionField of the Follow_Up message associated with the Sync message prior to retransmission of the Follow_Up message on the egress PTP Port. With the exception of the correctionField, the common header of the Follow_Up message shall be identical to the common header of the received Follow_Up message. The egress PTP Port shall make any needed corrections to checksums or other content dependent fields of the PTP message.

NOTE 1—To correctly associate Sync and Follow_Up messages requires that the Transparent Clock maintain a record of the sourcePortIdentity and sequenceId fields of the Sync message for comparison with the sourcePortIdentity and sequenceId fields of Follow_Up messages. The data type of the correctionField allows the <residenceTime> and the <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

NOTE 2—The data type of the correctionField allows the <residenceTime>, <meanLinkDelay> and the <delayAsymmetry> to be expressed to a fraction of a nanosecond if this accuracy is supported by the Transparent Clock.

10.3.3 Alternate processing of Sync and Follow_Up messages on peer-to-peer Transparent Clocks

10.3.3.1 General requirements

When processing Sync messages, corrections for asymmetry, residenceTime, and <meanLinkDelay> may be implemented in ingress PTP Ports, egress PTP Ports, the PTP Common Core, or any combination and using any of the available timestamp and correction fields subject to the other restrictions of 10.3.3.2.

NOTE—The above provides a context for implementations to have more flexibility in meeting the terms of 10.3.3.2. Egress PTP Ports need to have access to the following information to fulfill these terms:

- Which corrections have been done in the ingress PTP Port, the PTP Common Core or the information to make the corrections in the egress PTP Port.
- The value of the applicable ingress timestamp or a specification on how to correct for <residenceTime> based on actions by the ingress PTP Port.

In all cases, the twoStepFlag of a Sync message received at egress PTP Port on the Transparent Clock from the ingress PTP Port of the PTP Instance indicates whether or not an associated Follow_Up message will be received from the ingress PTP Port.

If the twoStepFlag of Sync messages received by the egress PTP Port is FALSE, then:

- a) If the egress PTP Port is a one-step PTP Port, then after implementing the corrections of 10.3.3.2, it shall retransmit the received PTP message,
- b) If the egress PTP Port is a two-step PTP Port, then after implementing the corrections of 10.3.3.2, it shall retransmit the received PTP message and generate and transmit an associated Follow_Up.

If the twoStepFlag of Sync message received by the egress PTP Port is TRUE, then:

- c) If the egress PTP Port is a one-step PTP Port, it shall wait for the associated Follow_Up message from the ingress PTP Port of the Transparent Clock and then after implementing the corrections of 10.3.3.2, retransmit the Sync message.
- d) If the egress PTP Port is a two-step PTP Port, it shall, after implementing the corrections of 10.3.3.2:
 - 1) Retransmit the Sync message,
 - 2) Wait for and then retransmit the associated Follow_Up message from the ingress PTP Port of the Transparent Clock.

In all cases, the twoStepFlag for Sync messages transmitted from the egress PTP Port shall be TRUE if an associated Follow_Up message will follow and FALSE otherwise. For all other PTP message types, the twoStepFlag of PTP messages transmitted from the egress PTP Port shall be FALSE.

In all cases, the ingress and egress PTP Ports may make use of the messageTypeSpecific field of the common header as specified in 13.3.2.10.

In all cases, egress PTP Ports shall make any needed corrections to checksums or other content-dependent fields of the PTP message leaving the egress PTP Port.

In 10.3.3.2, the constraint notation <ingress/egress>[<Message type-1> (<fieldname>) + <Message type-2>] indicates:

- e) Whether the PTP messages involved are those received by an ingress PTP Port or transmitted or retransmitted by an egress PTP Port of the Transparent Clock.
- f) The name of the PTP message, for example, Sync, and the name of the field of the named PTP message, for example, originTimestamp.

10.3.3.2 Constraints on processing of Sync and Follow_Up messages

In the following constraint specifications, ingress twoStepFlag and egress twoStepFlag refer to the twoStepFlag of a Sync message received on an ingress PTP Port and the corresponding Sync message retransmitted on the egress PTP Port.

The <residenceTime> is that of the Sync message.

The <meanLinkDelay> is that of the PTP Link on the ingress PTP Port receiving the Sync message.

The following constraints shall be satisfied for the respective cases.

10.3.3.2.1 All cases

Except for the twoStepFlag and the correctionField, the common header of Sync messages shall not be changed from ingress to egress. Except for the twoStepFlag, which shall be FALSE, the correctionField, and the messageType, which shall be set per 13.3.2.3, the common header of an egress Follow_Up message, if present, shall be identical to the common header of the associated Sync message.

10.3.3.2.2 Ingress and egress twoStepFlags both FALSE

$$\begin{aligned} &\text{Egress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] = \text{ingress} [\text{Sync}(\text{originTimestamp}) \\ &+ \text{Sync}(\text{correctionField})] + \text{ingress} [\text{<delayAsymmetry>} + \text{<residenceTime>} + \\ &\text{<meanLinkDelay>} \end{aligned}$$

10.3.3.2.3 Ingress twoStepFlag FALSE and egress twoStepFlag TRUE

$$\begin{aligned} &\text{Egress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correction} \\ &\text{field})] = \text{ingress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] + \text{ingress} \\ &[\text{<delayAsymmetry>} + \text{<residenceTime>} + \text{<meanLinkDelay>} \end{aligned}$$

10.3.3.2.4 Ingress twoStepFlag TRUE and egress twoStepFlag FALSE

$$\begin{aligned} \text{Egress} [\text{Sync}(\text{originTimestamp}) + \text{Sync}(\text{correctionField})] = & \text{ingress} \\ & [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correctionField})] \\ & + \text{ingress} [<\text{delayAsymmetry}>] + <\text{residenceTime}> + <\text{meanLinkDelay}> \end{aligned}$$

10.3.3.2.5 Ingress twoStepFlag TRUE and egress twoStepFlag TRUE

$$\begin{aligned} \text{Egress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) + \text{Sync}(\text{correctionField}) \\ + \text{Follow\_Up}(\text{correctionField})] = & \text{ingress} [\text{Follow\_Up}(\text{preciseOriginTimestamp}) \\ + \text{Sync}(\text{correctionField}) + \text{Follow\_Up}(\text{correctionField})] + & \text{ingress} [<\text{delayAsymmetry}>] \\ + <\text{residenceTime}> + <\text{meanLinkDelay}> \end{aligned}$$

11. Clock offset, path delay, residence time, and asymmetry corrections

11.1 General specifications

The specifications in Clause 11 provide mechanisms for conveying timestamps generated at the sources of PTP event messages along with any corrections needed to ensure that the recipient of the PTP event message receives the most accurate timestamp possible. The actual distribution of the time information between the originTimestamp or the preciseOriginTimestamp and the correctionField fields is implementation dependent, providing the distribution shall be such that a receiving PTP Instance performing the computations on timestamp and correctionField fields, as specified in the following subclauses, obtains the most accurate timestamp possible.

Clause 11 specifies:

- a) The computation of the offset in time between a Slave Clock and a Master Clock, that is, <offsetFromMaster>.
- b) The delay request-response mechanism: This mechanism measures the <meanPathDelay> between a pair of PTP Ports, each of which supports the state machine of 9.2.5.
- c) The peer-to-peer delay mechanism: This mechanism measures the <meanLinkDelay> between a pair of PTP Ports, each of which supports the peer-to-peer delay mechanism.
- d) The correction for <meanLinkDelay> in peer-to-peer Transparent Clocks.
- e) The use of Special Ports in Boundary Clocks and Ordinary Clocks.
- f) The correction of PTP timing messages in Transparent Clocks based on the measurement of the residence time within a Transparent Clock.
- g) Corrections for asymmetry.

NOTE 1—The recommendation that the timestamps themselves be the best possible estimate of the time enables simple PTP Instances that only need approximate time to ignore the correctionField.

NOTE 2—The latitude in the distribution of time between the timestamp and the correctionField allows flexibility in the PTP Instance design. For example, a PTP Instance might generate an approximate time when a PTP message is assembled and insert the resulting required correction into the appropriate correctionField. An unavoidable circumstance is the representation of fractional nanoseconds. Fractional nanoseconds cannot be represented in the Timestamp data type and are transmitted in a correctionField, leaving it to the receiving PTP Instance to combine the two to get the actual timestamp.

11.2 Computation of <offsetFromMaster> in Ordinary Clocks and Boundary Clocks

The time error between a Slave Clock and a Master Clock is defined as follows:

$$\text{<offsetFromMaster>} = \text{<Time on the Slave Clock>} - \text{<Time on the Master Clock>}$$

where all times are measured at the same instant.

NOTE—The quantity <offsetFromMaster> above reflects the status before the Local PTP Clock is synchronized.

If the PTP Port in the SLAVE state is not a Special Port, and it implements the delay request-response or peer-to-peer delay mechanism, the Slave PTP Instance shall do the following:

- a) Upon receipt of a Sync message, the Slave PTP Instance generates a timestamp t_2 , the <syncEventIngressTimestamp>, corrected for latency per 7.3.4. The Slave PTP Instance also computes the <correctedSyncCorrectionField> by adding the value of the ingress path <delayAsymmetry> (see 7.4.2) to the correctionField of the received Sync. No correction is made for egress path <delayAsymmetry>.
- b) If the twoStepFlag bit of the flagField of the of the Sync message is FALSE, indicating that a Follow_Up message will not be received, then the Slave PTP Instance computes <offsetFromMaster> as follows:

$$\begin{aligned} \text{<offsetFromMaster>} &= \text{<syncEventIngressTimestamp>} - \text{<originTimestamp>} \\ &- \text{<meanDelay>} - \text{<correctedSyncCorrectionField>} \end{aligned}$$

- c) If the twoStepFlag bit of the flagField of the of the Sync message is TRUE, indicating that a Follow_Up message will be received, then the Slave PTP Instance computes <offsetFromMaster> as follows:

$$\begin{aligned} \text{<offsetFromMaster>} &= \text{<syncEventIngressTimestamp>} - \text{<preciseOriginTimestamp>} \\ &- \text{<meanDelay>} - \text{<correctedSyncCorrectionField>} - \text{correctionField of Follow\_Up} \\ &\text{message} \end{aligned}$$

Where

- 1) The <originTimestamp> shall be the value of the originTimestamp field in the received Sync message.
- 2) The <preciseOriginTimestamp> shall be the value of the preciseOriginTimestamp field in the received Follow_Up message.
- 3) The <meanDelay> shall be
 - i) The value of the <meanPathDelay> specified in 11.3 if the PTP Port is configured to use the delay request-response mechanism, or
 - ii) The value of the <meanLinkDelay> specified in 11.4 if the PTP Port is configured to use the peer-to-peer delay mechanism, or
 - iii) Zero if the PTP Port is configured not to use either path delay mechanism (see 8.2.2.4).

The value of <offsetFromMaster> shall be stored in the member currentDS.offsetFromMaster.

If the PTP Port in the SLAVE state is a Special Port, the <offsetFromMaster> is computed as described in 11.5.3.3.

11.3 Delay request-response mechanism for Ordinary Clocks and Boundary Clocks

11.3.1 Delay request-response mechanism general requirements

The delay request-response mechanism measures the $\langle \text{meanPathDelay} \rangle$ between a pair of PTP Ports, each of which supports the state machine of 9.2.5. The delay request-response mechanism uses the PTP messages Sync, Delay_Req, Delay_Resp, and possibly Follow_Up as shown in the timing diagram of Figure 41. This mechanism shall be executed independently in each supported domain of a PTP Node.

The timestamps t_1 and t_2 for the Sync message and t_3 and t_4 for the Delay_Req message of Figure 41 shall be generated as defined in 7.3.4.2. Timestamps t_1 and t_4 shall be generated using the time of the Master PTP Instance, and the timestamps t_2 and t_3 shall be generated using the time of the Slave PTP Instance.

NOTE—The nominal value of the $\langle \text{meanPathDelay} \rangle$ is computed as $\langle \text{meanPathDelay} \rangle = [(t_2 - t_1) + (t_4 - t_3)]/2 = [(t_2 - t_3) + (t_4 - t_1)]/2$

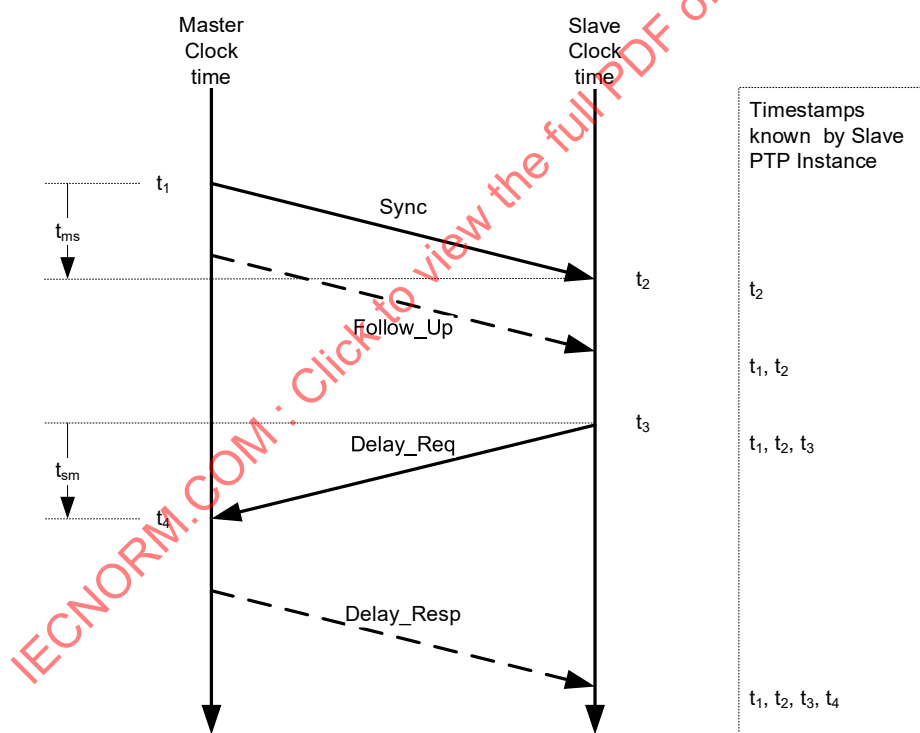


Figure 41—Delay request-response path length measurement

11.3.2 Delay request-response mechanism operational specifications

The actual value of the $\langle \text{meanPathDelay} \rangle$ is measured and computed by the PTP Port in the PTP SLAVE state as follows between each pair of PTP Ports in the MASTER and SLAVE states:

- a) A PTP Port in the MASTER state prepares and issues Sync messages per 9.5.9 and Clause 13. The PTP Port shall generate an egress timestamp t_1 . If the PTP Port is a two-step PTP Port, it also prepares and issues Follow_Up messages per 9.5.10 and Clause 13.
- b) Upon receipt of a Sync message, the PTP Port in the SLAVE state generates t_2 , and computes $\langle \text{correctedSyncCorrectionField} \rangle$ per item a) of 11.2.
- c) If required to send a Delay_Req message based on the requirements of 9.5.11, the PTP Port in the SLAVE state shall:
 - 1) Prepare a Delay_Req message per 13 with the correctionField (see 13.3.2.9) set to 0.
 - 2) Set the originTimestamp to 0 or an estimate no worse than ± 1 s of the egress time of the Delay_Req message.
 - 3) Subtract the value of the egress path $\langle \text{delayAsymmetry} \rangle$ from the correctionField of the Delay_Req message prior to transmission. No correction shall be made for ingress path $\langle \text{delayAsymmetry} \rangle$.
 - 4) Send the Delay_Req message and generate and save timestamp t_3 .
- d) Upon receipt of the Delay_Req message, the PTP Port in the MASTER state shall:
 - 1) Generate timestamp t_4 .
 - 2) Prepare a Delay_Resp message as follows:
 - i) Copy the common header of the Delay_Req to the header of the Delay_Resp message with the exceptions listed below.
 - ii) Set flagField twoStepFlag bit to FALSE.
 - iii) Set the messageType per 13.3.2.
 - iv) Set the sourcePortIdentity set per 13.3.2.11.
 - v) Copy the sourcePortIdentity field of the Delay_Req message to the requestingPortIdentity field of the Delay_Resp message.
 - vi) Set the correctionField of the Delay_Resp message to 0.
 - vii) Add the correctionField of the Delay_Req message to the correctionField of the Delay_Resp message.
 - viii) Set the receiveTimestamp field of the Delay_Resp message to the seconds and nanoseconds portion of the time t_4 .
 - ix) Subtract any fractional nanosecond portion of t_4 from the correctionField of the Delay_Resp message.
 - 3) Issue the Delay_Resp message based on the requirements of 9.5.12.
- e) Upon receipt of the Delay_Resp message by the PTP Port in the SLAVE state:
 - 1) If the received Sync message indicated that a Follow_Up message will not be received, the $\langle \text{meanPathDelay} \rangle$ shall be computed as:

$$\langle \text{meanPathDelay} \rangle = [(t_2 - t_3) + (\text{receiveTimestamp of Delay\_Resp message} - \text{originTimestamp of Sync message}) - \langle \text{correctedSyncCorrectionField} \rangle - \text{correctionField of Delay\_Resp message}] / 2$$
 - 2) If the received Sync message indicated that a Follow_Up message will be received, the $\langle \text{meanPathDelay} \rangle$ shall be computed as:

$$\langle \text{meanPathDelay} \rangle = [(t_2 - t_3) + (\text{receiveTimestamp of Delay\_Resp message} - \text{preciseOriginTimestamp of Follow\_Up message}) -$$

$\langle \text{correctedSyncCorrectionField} \rangle$ – correctionField of Follow_Up message –
 correctionField of Delay_Resp message]/2

The value of the $\langle \text{meanPathDelay} \rangle$ shall be stored in the member `currentDS.meanDelay`.

NOTE—The delay request-response path length measurement normally uses the timestamps and correctionField of the most recent Sync and corresponding Follow_Up message prior to the Delay_Req message. However, this measurement can use any Sync and corresponding Follow_Up message, although this will reduce the accuracy of the computed $\langle \text{meanPathDelay} \rangle$.

11.4 Peer-to-peer delay mechanism

11.4.1 Peer-to-peer delay mechanism general requirements

If the Common Mean Link Delay Service option of 16.6 is used, the specifications of 11.4.1, 11.4.2, and 11.4.3 shall not apply except as indicated in 16.6.

The peer-to-peer delay mechanism measures the PTP Port-to-PTP Port propagation time, that is, the $\langle \text{meanLinkDelay} \rangle$, between two communicating PTP Ports supporting the peer-to-peer delay mechanism.

NOTE 1—The peer-to-peer delay mechanism for PTP Ports using a Timestamping Clock that is not the Local PTP Clock (e.g., as will happen in PTP Ports of a nonsynchronized Transparent Clock) is basically the same as for PTP Ports using the Local PTP Clock as Timestamping Clock. However, when using a nonsynchronized Timestamping Clock, then the $\langle \text{meanLinkDelay} \rangle$ measurement (if no further correction is applied) will deviate from the desired measurement due to the syntonization error (see 6.6.6).

This measurement should be made on all PTP Ports of a PTP Instance including those that are blocked by lower level protocols. The addressing or other mechanisms to support this requirement are network specific and are specified in the relevant annexes to this standard.

The $\langle \text{meanLinkDelay} \rangle$ measurement shall be made independently by each PTP Port implementing the peer-to-peer delay mechanism.

NOTE 2—This requirement means that the $\langle \text{meanLinkDelay} \rangle$ is known by PTP Ports on both ends of a PTP Link. This allows path delay corrections to be made immediately upon reconfiguration of the network.

For Boundary Clocks and Ordinary Clocks, the peer-to-peer delay mechanism shall be independent of whether the PTP Port is in the MASTER or the SLAVE state.

The peer-to-peer delay mechanism uses the PTP messages `Pdelay_Req`, `Pdelay_Resp`, and possibly `Pdelay_Resp_Follow_Up`, as shown in the timing diagram of Figure 42.

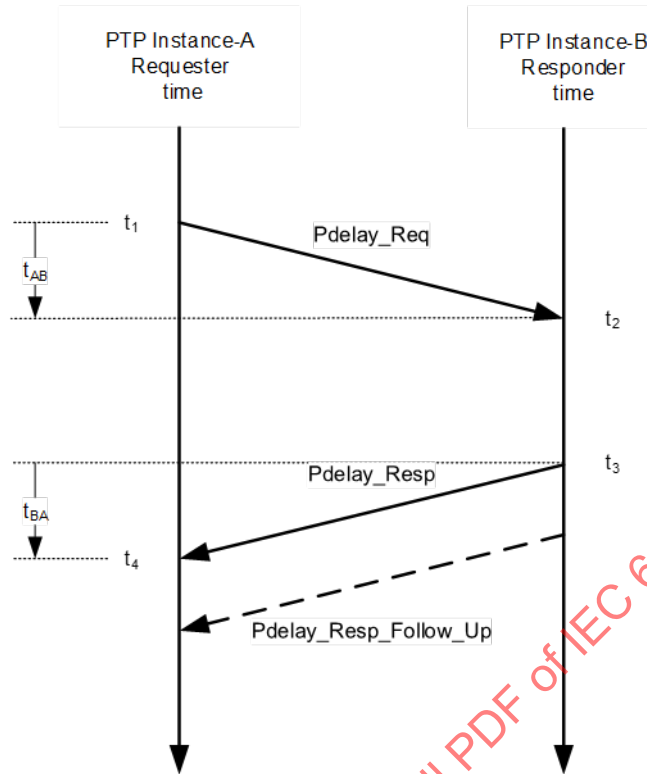


Figure 42—Peer-to-peer delay link measurement

The timestamps t_1 and t_2 for the Pdelay_Req message and t_3 and t_4 for the Pdelay_Resp message of Figure 42 are measured as defined in 7.3.4.

Timestamps t_1 and t_4 shall be generated by PTP Instance-A (the requester) using the Timestamping Clock of PTP Instance-A as specified in 7.3.4.3.

Timestamps t_2 and t_3 shall be generated by PTP Instance-B (the responder) using the Timestamping Clock of PTP Instance-B as specified in 7.3.4.3.

NOTE 3—The nominal value of the <meanLinkDelay> is computed as

$$\langle \text{meanLinkDelay} \rangle = [(t_2 - t_1) + (t_4 - t_3)]/2 = [(t_2 - t_3) + (t_4 - t_1)]/2$$

The actual value is specified in 11.4.2.

11.4.2 Peer-to-peer delay mechanism operational specifications

Except when using option 16.10, in which case the modifications of this subclause specified in 16.10 shall hold, the actual value of the <meanLinkDelay> is computed as follows for each instance of a peer-to-peer delay measurement between PTP Ports:

- a) If required to send a Pdelay_Req message based on the requirements of 9.5.13, the requester PTP Port on PTP Instance-A prepares a Pdelay_Req message as follows:
 - 1) The correctionField (see 13.3.2.9) shall be set to 0.
 - 2) The domainNumber field of the header should be set to the domain of PTP Instance-A.

- 3) Prior to transmission on an egress PTP Port, the correctionField of the transmitted Pdelay_Req message shall be modified by subtracting the value of the egress path <delayAsymmetry> from the correctionField of the transmitted Pdelay_Req message.
 - 4) The originTimestamp shall be set to 0 or an estimate no worse than ± 1 s of the egress timestamp, t_1 , of the Pdelay_Req message.
 - 5) PTP Instance-A shall send the Pdelay_Req message and generate and save timestamp t_1 .
- b) If the responder PTP Port on PTP Instance-B is a one-step PTP Port, it shall:
- 1) Generate timestamp t_2 upon receipt of the Pdelay_Req message.
 - 2) Prepare a Pdelay_Resp message with the common header of the Pdelay_Resp message as specified in 13.3.2 except for
 - i) The sequenceId field, which is the sequenceId field of the received Pdelay_Req message, and
 - ii) The correctionField, which is the correctionField of the received Pdelay_Req message.
 - 3) Copy the sourcePortIdentity field from the Pdelay_Req message to the requestingPortIdentity field of the Pdelay_Resp message.
 - 4) Then:
 - i) Set to 0 the requestReceiptTimestamp field of the Pdelay_Resp message.
 - ii) Issue the Pdelay_Resp message and generate timestamp t_3 upon sending based on the requirements of 9.5.14.
 - iii) After t_3 is generated but while the Pdelay_Resp message is leaving the responder, add the turnaround time $t_3 - t_2$ to the correctionField of the Pdelay_Resp message and make any needed corrections to checksums or other content-dependent fields of the Pdelay_Resp message.

NOTE—The data type of the correctionField allows the time interval $t_3 - t_2$ to be expressed to a fraction of a nanosecond if needed and this accuracy is supported by the responder.

- c) If the delay responder is a two-step PTP Port, it shall:
- 1) Generate timestamp t_2 upon receipt of the Pdelay_Req message.
 - 2) Prepare a Pdelay_Resp and a Pdelay_Resp_Follow_Up message with the common header of the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively, as specified in 13.3.2, except for the sequenceId, correctionField, and domainNumber fields, which are as specified in the subsequent points.
 - 3) Copy the correctionField from the Pdelay_Req message to the correctionField of the Pdelay_Resp_Follow_Up message, and set correctionField of the Pdelay_Resp message to 0.
 - 4) Copy the sequenceId field from the Pdelay_Req message to the sequenceId field of the Pdelay_Resp and the Pdelay_Resp_Follow_Up messages.
 - 5) Copy the sourcePortIdentity field from the Pdelay_Req message to the requestingPortIdentity field of the Pdelay_Resp and the Pdelay_Resp_Follow_Up messages.
 - 6) Copy the domainNumber field from the Pdelay_Req message to the domainNumber field of the Pdelay_Resp and Pdelay_Resp_Follow_Up messages.
 - 7) Execute either Option A or Option B:

Option A:

 - i) Set to 0 the requestReceiptTimestamp fields of the Pdelay_Resp message.
 - ii) Issue the Pdelay_Resp message and generate timestamp t_3 upon sending based on the requirements of 9.5.14.
 - iii) In the Pdelay_Resp_Follow_Up message, set the responseOriginTimestamp field to 0, and add the turnaround time $t_3 - t_2$ to the correctionField.
 - iv) Issue the Pdelay_Resp_Follow_Up message based on the requirements of 9.5.15.

Option B:

- i) In the Pdelay_Resp message, set the requestReceiptTimestamp field to the seconds and nanoseconds portion of the time t_2 , and subtract any fractional nanosecond portion of t_2 from the correctionField.
- ii) Issue the Pdelay_Resp message and generate timestamp t_3 upon sending based on the requirements of 9.5.14.
- iii) In the Pdelay_Resp_Follow_Up message, set the responseOriginTimestamp field to the seconds and nanoseconds portion of the time t_3 , and add any fractional nanosecond portion of t_3 to the correctionField.
- iv) Issue the Pdelay_Resp_Follow_Up message based on the requirements of 9.5.15.
- d) The requester PTP Port on PTP Instance-A, upon receiving a Pdelay_Resp message shall:
 - 1) Generate timestamp t_4 upon receipt of the Pdelay_Resp message.
 - 2) To correct for asymmetry of the PTP Link connected to the ingress PTP Port, compute the <correctedPdelayRespCorrectionField> by adding the value of the ingress <delayAsymmetry> to the correctionField of the received Pdelay_Resp message.
 - 3) If the twoStepFlag of the received Pdelay_Resp message is FALSE, indicating that no Pdelay_Resp_Follow_Up message will be received, compute the <meanLinkDelay> as:

$$\text{<meanLinkDelay>} = [(t_4 - t_1) - \text{<correctedPdelayRespCorrectionField>}] / 2$$
 - 4) If the twoStepFlag of the received Pdelay_Resp message is TRUE indicating that a Pdelay_Resp_Follow_Up will be received, compute the <meanLinkDelay> as follows:

$$\begin{aligned} \text{<meanLinkDelay>} = & [(t_4 - t_1) - (\text{responseOriginTimestamp} \\ & - \text{requestReceiptTimestamp}) - \text{<correctedPdelayRespCorrectionField>} \\ & - \text{correctionField of Pdelay_Resp_Follow_Up}] / 2 \end{aligned}$$

The responder PTP Port on PTP Instance-B, upon receipt of a Pdelay_Req message, shall issue the associated Pdelay_Resp message as quickly as possible, consistent with the other requirements of this subclause, to minimize the turnaround time, $(t_3 - t_2)$.

For Boundary Clocks and Ordinary Clocks, the computed value of the <meanLinkDelay> shall be stored

- e) For all peer-to-peer ports, in the portDS.meanLinkDelay, and
- f) For the peer-to-peer port in the PTP SLAVE state, if present in the PTP Instance, in the currentDS.meanDelay.

For Transparent Clocks, the computed value of the <meanLinkDelay> shall be stored in

- g) transparentClockPortDS.meanLinkDelay, if it is implemented (see 8.1.1.3 and 8.3.1.1), or
- h) portDS.meanLinkDelay, if it is implemented (see 8.3.1.2 and 8.2.15.3.3), or
- i) In a way that is implementation specific, if neither transparentClockPortDS.meanLinkDelay nor portDS.meanLinkDelay is implemented.

11.4.3 Restriction on the use of the peer-to-peer delay mechanism

A requester PTP Instance-A, might receive 0, 1, or multiple Pdelay_Resp messages for each transmitted Pdelay_Req.

Multiple responses can be detected by observing that the sourcePortIdentity field of the Pdelay_Resp messages differ.

NOTE—For example, multiple responses can occur if there is an ordinary bridge or other similar multicast and multiport devices between PTP Instance-A and multiple PTP Instance-B devices, each of which is attached to a peer-to-peer Transparent Clock (see Figure 43). Although the multiple Pdelay_Resp responses can be distinguished, there is no mechanism in this standard that allows the PTP Link delay associated with each of the responses from the multiple Transparent Clocks to be correctly assigned to a received Sync message. Thus, in the example of Figure 43, PTP Instance-A receives Pdelay_Resp messages from both Transparent Clocks M and S and can compute the respective PTP Link delay. However, when receiving a Sync message from PTP Instance-B-M via Transparent Clock-M, the sourcePortIdentity field value will be that of PTP Instance-B-M and will not match the sourcePortIdentity fields of the Pdelay_Resp messages from either of the two Transparent Clocks.

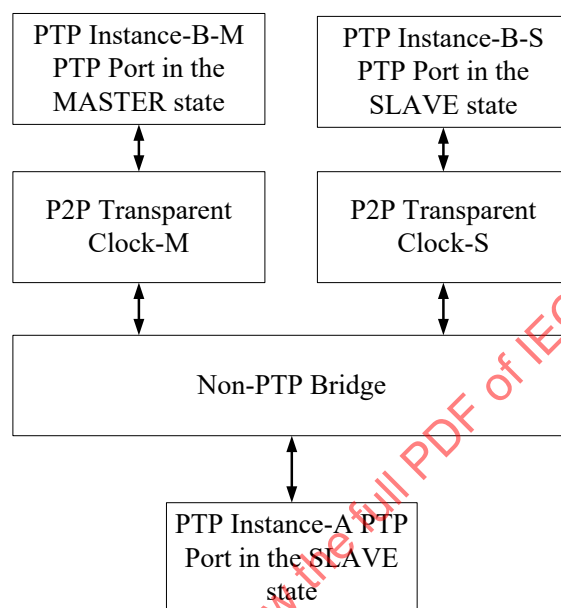


Figure 43—Informative example of a problematic peer-to-peer topology

The following actions should be taken in these cases:

- a) When no Pdelay_Resp is received, PTP Instance-A should periodically transmit a Pdelay_Req message to check for the appearance of a PTP Instance-B. The transmission rate in this case is implementation or profile specific. The implementation- or profile-specific rate should be established using the mechanisms of 7.7.2.1 and 7.7.2.5.
- b) When a single Pdelay_Resp is received, the protocol of 11.4.2 should be executed as specified.
- c) When multiple Pdelay_Resp messages are received, PTP Instance-A shall either:
 - 1) Enter the FAULTY state if an Ordinary Clock or Boundary Clock, or a fault condition if a peer-to-peer Transparent Clock. In this case, the PTP Instance may periodically transmit a Pdelay_Req message to check for the resolution of this condition. The transmission rate in this case is implementation specific. In this case, Sync and Follow_Up messages received on the PTP Port shall be discarded, or
 - 2) Take implementation-specific measures to resolve the issue.

11.5 MDMI interface and Special Ports

11.5.1 General requirements

When standards for physical media specify the Special PTP MD Adapter of 11.5, implementation of Special Ports according to this standard shall interact with the media using the MDMI interface, and the Special PTP MD Adapter shall conform to standards for that physical media.

Implementations of E2E PTP Ports and P2P PTP Ports using these media must use PTP messages following the specifications of Clause 10 through Clause 12.

Special Ports shall be used to transfer time within the domain but over a network where the time transfer is not based on the use of PTP timing messages as specified in Clause 9 and Clause 10, 11.1 through 11.4, and Clause 13.

The model of a Special Port is illustrated in Figure 44.

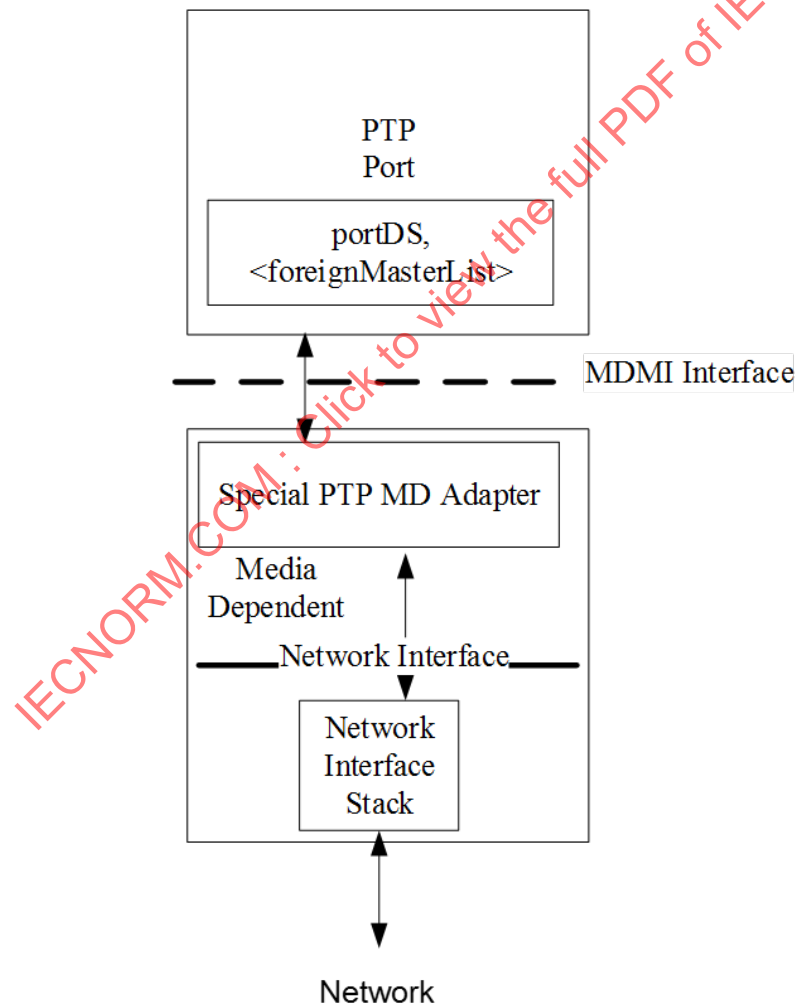


Figure 44—Special Port model

Special Ports require that the connecting network supports media-specific methods for:

- a) Accepting time, t_1 , from a Special Port on a PTP Instance, and
- b) Measuring the path delay, PD, between the Special Port providing time and the Special Port accepting time, and
- c) Delivering time, $t_1 + PD$, to the accepting Special Port on the receiving PTP Instance, and
- d) Providing a means to transport in either direction, PTP nontiming messages

Special Ports shall be used only on Ordinary Clocks or Boundary Clocks.

NOTE—The PTP Instance hosting the Special Ports either needs to participate in the BMCA so that the PTP state of each Special Port determines the direction of time transfer, or the direction of time transfer needs to be fixed by configuration of the Special Ports, for example, by configuring one to be masterOnly or by using the option of 17.6.

11.5.2 Operation of the components in a Special Port

11.5.2.1 Nontiming PTP message handling by a Special Port

PTP nontiming messages are handled by a Special Port in the same way as in an E2E or a P2P PTP Port.

On the MD side, the MD Adapter block encapsulates Announce, Signaling, and PTP management message information according to the rules of the transport; for example, for Ethernet, the Ethernet header is prepended and the checksum appended. On transmission, the MD Adapter block passes the encapsulated message information to the Network Interface Stack. On reception, the MD Adapter block accepts the encapsulated message information from the Network Interface Stack and retrieves the PTP portion of the frame; for example, for Ethernet, the Ethernet header and checksum are removed.

11.5.2.1.1 Announce messages

On transmission, Announce message information transmission is initiated on the MI side of the MDMI interface by the PTP Port block, as required by 9.5.8. The Announce message information as specified in 13.5 is passed across the MDMI interface.

On reception, the PTP frame is passed across the MDMI interface. The PTP Port block handles this information as required by 9.5.3.

11.5.2.1.2 Signaling messages

On transmission, Signaling message information transmission is initiated on the MI side of the MDMI interface by the PTP Port block, as required by 13.12.2. The Signaling message information as specified in 13.12.2 is passed across the MDMI interface.

On reception, the PTP frame is then passed across the MDMI interface. The PTP Port block handles this information as required by 13.12.1.

11.5.2.1.3 PTP management messages

On transmission, PTP management message information transmission is initiated on the MI side of the MDMI interface by the PTP Port block, as required by the respective subclauses of Clause 15 through Clause 17. The PTP management message information as specified in Clause 15 is passed across the MDMI interface.

On reception, the PTP frame is then passed across the MDMI interface. The PTP Port block handles this information as required by the respective subclauses of Clause 15 through Clause 17.

11.5.2.2 Time transfer by a Special Port

11.5.2.2.1 PTP Port block

The PTP Port block handles all interactions with the Special PTP MD Adapter of the Special Port. If the PTP Port state is MASTER, the PTP Port generates the sending of time synchronization information. The successive times that this information is sent are specified in 9.5.9.2. The information is transmitted to the MD Adapter for the PTP Port in an MDSyncSend structure. The members of the MDSyncSend structure are set as described in 11.5.3.2.

If the PTP Port state is SLAVE, the PTP Port transmits the information received in an MDSyncReceive structure to the PTP Common Core block (see 6.5.2.1.3). The PTP Common Core block uses the information received from the PTP Port in the SLAVE state to synchronize and possibly syntonize the Local PTP Clock to the Grandmaster Clock.

11.5.2.2.2 MDMI interface general properties

The MDMI interface separates the PTP Port and MD Adapter blocks. The information flowing across this interface in general depends on the state of the PTP Port but is the same for each PTP Port type. Information received by the PTP Instance on the PTP Port in the SLAVE state is transferred across the MDMI interface in an MDSyncReceive structure, from the Special PTP MD Adapter block to the PTP Port block (see 11.5.3.1). Information transmitted by the PTP Instance on PTP Port in the MASTER state is transferred across the MDMI interface in an MDSyncSend structure, from the PTP Port block to the Special PTP MD Adapter block (see 11.5.3.2). If the PTP Port is in the PASSIVE state, no timing information is transferred across the MDMI interface.

This interface is new to this edition and is introduced to facilitate the use within the PTP protocol of media that implement their own methodology for measuring path delay and transferring time, for example, IEEE Std 802.11.

11.5.2.2.3 Special PTP MD Adapter block

The Special PTP MD Adapter block measures mean path delay; receives time synchronization information from the PTP Port at the other end of the attached link if the PTP Port is a PTP Port in the SLAVE state; sends respective information across the MDMI interface via an MDSyncReceive structure; receives information sent across the MDMI interface via an MDSyncSend structure; and sends the respective time synchronization information to the PTP Port at the other end of the attached link. The mean path delay measurement is done using methods specific to the respective medium. The transport of time synchronization information to the PTP Port at the other end of the attached link is done using methods and messages specific to the respective medium.

The subclauses of IEEE Std 802.1AS-2011 referenced below are the only subclauses of IEEE Std 802.1AS-2011 that are normative for the Special PTP MD Adapter block.

The Special PTP MD Adapter block for IEEE 802.11 transport shall meet the requirements in the following subclauses of Clause 12 of IEEE Std 802.1AS-2011: 12.1.1, which describes the Timing Measurement; 12.1.2, which describes the layering; 12.4.1, which describes the master state machine; 12.4.2, which describes the slave state machine; and 12.5, which describes the carrying of information needed by the PTP protocol.

The Special PTP MD Adapter block for the IEEE 802.3 EPON transport shall meet the requirements in the following subclauses of Clause 13 of IEEE Std 802.1AS-2011: 13.1.2 and 13.1.4, which describe the procedure for time transport in EPON; subclause 13.5, which describes the layering; 13.8.1, which describes the master (EPON requester) state machine (and uses information described in 13.7); 13.8.2, which describes the slave state machine (EPON responder); and 13.2, 13.3, and 13.6, which describe the carrying of information needed by the PTP protocol.

The Special PTP MD Adapter block for Coordinated Shared Network (CSN) transport shall meet the requirements of the following subclauses of Annex E of IEEE Std 802.1AS-2011: E.2 and E.4, which describe time transport in a CSN E.3, which describes the layering; and E.5, which describes the carrying of information needed by the PTP protocol.

The following two modes are for time transport over a CSN:

- a) Time transport where a network reference clock (i.e., a CSN network reference clock) is not present, and
- b) Time transport where a network reference clock is present

In mode a), the peer-to-peer delay mechanism is used to measure mean link delay, and the MD Adapter block is a PTP P2P MD Adapter in this case. In mode b), the local clocks of the CSN nodes are synchronized to each other but not necessarily to the Local PTP Clock of any of the CSN devices. Time is transferred across the CSN using Sync and, in the two-step case, Follow_Up messages. An incoming Sync message (at the ingress CSN device) is timestamped relative to the CSN network reference clock. The CSN TLV, described in E.5.2.1.1 of IEEE Std 802.1AS-2011, is appended to the Sync or, in the two-step case, Follow_Up message. The Sync and Follow_Up messages are transmitted across the CSN to the egress CSN device. The outgoing Sync message at the respective Master PTP Instance PTP Port of the egress CSN device is timestamped, and the fields of the Sync, and in the two-step case, Follow_Up message are computed using the <syncEventEgressTimestamp> and information contained in the CSN TLV (the CSN TLV is not appended to any PTP messages that are transmitted from the egress CSN device).

Specific CSNs include Multi-Media Over Coax (MoCA) and ITU-T G.hn ITU-T G.9960. Specific normative requirements for these CSNs are given in E.6 of IEEE Std 802.1AS-2011.

11.5.2.2.4 Network Interface Stack

The Network Interface Stack includes the layers of the underlying transport network. The transport network provides the transport for all information, including PTP Communication. The Network Interface Stack is separated from the Special PTP MD Adapter by the Network Interface. The specification of the Network Interface Stack and Network Interface is outside of PTP.

NOTE—The transport of PTP Communication will not necessarily use all the layers of the transport network. The specific layers used depends on the transport. For example, the transport over an IEEE 802 bridged LAN uses layers 1 and 2.

11.5.3 Timing information transfer across the MDMI Interface

NOTE 1—In this subclause, the term “upstream PTP Instance” refers to the PTP Instance at the other end of the PTP Communication Path of the SLAVE (i.e., ingress) PTP Port of the PTP Instance being referenced or discussed. The term “downstream PTP Instance” refers to the PTP Instance at the other end of the PTP Communication Path of the MASTER (i.e., egress) PTP Port of the PTP Instance being referenced or discussed.

NOTE 2—Figure 45 illustrates a network configuration that is used in two running examples below. In both examples, Boundary Clocks BC-C and BC-D communicate via an IEEE 802.11 link attached to Special Ports, with the Special Port on BC-C in the MASTER state and the Special Port on BC-D in the SLAVE state. Also, in both examples,

BC-A is upstream of BC-C, and the PTP Communication Path between BC-A and BC-C contains Transparent Clock TC-B; and BC-F is downstream of BC-D, and the PTP Communication Path between BC-D and BC-F contains TC-E. The details of the PTP Ports of BC-A, TC-B, TC-E, and BC-F are not shown; however, the PTP Port on BC-A that communicates with the SLAVE PTP Port of BC-C is in the MASTER state, and the PTP Port of BC-F that communicates with the MASTER PTP Port of BC-D is in the SLAVE state. All the PTP Instances in this figure might have other PTP Ports, but these are not shown.

NOTE 3—In the first example, referred to as “Example 1,” the PTP Communication Paths between BC-A and BC-C, and between BC-D and BC-F, use the delay request-response mechanism, and TC-B and TC-E are end-to-end Transparent Clocks. In this example, the cumulative frequency method is not used.

NOTE 4—In the second example, referred to as “Example 2,” the PTP Communication Paths between BC-A and BC-C, and between BC-D and BC-F, use the peer-to-peer delay mechanism, and TC-B and TC-E are peer-to-peer Transparent Clocks. In Example 2, the cumulative frequency method is used.

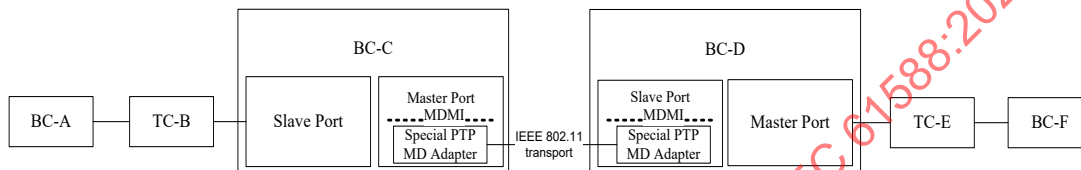


Figure 45—Example network—transport via Special Ports attached to an IEEE 802.11 link

11.5.3.1 Timing information transfer across the MDMI Interface on reception

The information transferred across the MDMI Interface, from the Special PTP MD Adapter block to the PTP Port block, is contained in the MDSyncReceive structure:

```
MDSyncReceive {
    domainNumber,
    sdoId,
    sourcePortIdentity,
    originOrPreciseOriginTimestamp,
    correctionField,
    logMessageInterval,
    upstreamTxTime,
    cumulativeRateRatio
}
```

NOTE 1—In the example of Figure 45, the MDSyncReceive information transfer occurs in the ingress PTP Port of BC-D.

NOTE 2—In IEEE Std 802.1AS-2011, the member cumulativeRateRatio is renamed “rateRatio.” Both are intended to provide the same information, namely, the value of the cumulativeRateRatio that needs to be distributed to downstream PTP Instances by the PTP Port in the MASTER state, for example, the Master port of BC-D in the example of Figure 45. For this example, the value is the <neighborRateRatio> (see 16.10.2) between the BC-D and the BC-C times the value of cumulateRateRatio received from BC-C. The value of rateRatio is determined by the Special PTP MD Adapter block in BC-D based on information specified in the IEEE Std 802.1AS-2011 for the supported transports (see 11.5.2.2.3).

NOTE 3—The information identified in MDSyncReceive and MDSyncSend are transported across the medium between the Special Port on the upstream and downstream PTP Ports based on TLVs specified in IEEE Std 802.1AS-2011 for each medium.

NOTE 4—The member rateRatio in IEEE Std 802.1AS-2011 has data type Double (i.e., Float64). However, the member is used in IEEE Std 802.1AS-2011 only internally, that is, in the MDSyncSend and MDSyncReceive structures, which are passed across the MDMI Interface. It is not used on-the-wire. It is stated in 6.3.1 of

IEEE Std 802.1AS-2011 that implementations are free to use any internal representation of data types if the internal representation does not change the semantics of any quantity visible via communications using the IEEE 802.1AS protocol or in the specified operations of the protocol.

The members of the MDSyncReceive structure are defined as follows.

11.5.3.1.1 domainNumber

The domainNumber is the number of the PTP domain on which the information was transmitted from the sending PTP Instance, for example, BC-C in the example of Figure 45.

11.5.3.1.2 sdoId

The sdoId is the sdoId associated with the domainNumber of the PTP domain on which the information was transmitted from the sending PTP Instance. It is either:

- a) The value specified-by-design, if permitted by 7.1.4, or
- b) The value defaultDS.sdoId of the sending PTP Instance

11.5.3.1.3 sourcePortIdentity

The sourcePortIdentity is the portIdentity of the PTP Port that sent this information.

11.5.3.1.4 originOrPreciseOriginTimestamp

The originOrPreciseOriginTimestamp is a value whose data type is Timestamp (see 5.3.3) such that, when combined with the correctionField (see 11.5.3.1.5), the result is the value of the Local PTP Clock of the upstream PTP Instance at an arbitrarily selected time X. The time X represents the same instant of time in 11.5.3.1.4, 11.5.3.1.5, and 11.5.3.1.7, and it is selected by the PTP MD Adapter block of the sending port of the upstream PTP Instance, that is, BC-C in Figure 45.

NOTE 1—In Example 1 and Example 2 illustrated by Figure 45, X is the time of the <syncEventEgressTimestamp> of the sending of this time synchronization information by the upstream PTP Instance (i.e., BC-C); that is, X represents the time that this time synchronization information was sent. In these examples, the value of originOrPreciseOriginTimestamp sent across the MDMI interface at BC-D is the time of X relative to the Local PTP Clock at BC-C, excluding any sub-nanosecond portion.

NOTE 2—If Example 2 were modified such that TC-B and TC-E were removed and the other PTP Instances followed the specifications of IEEE Std 802.1AS-2011, then the value of originOrPreciseOriginTimestamp sent across the MDMI interface at BC-D would be the time relative to the Grandmaster Clock that the time synchronization information received at BC-D was sent by the Grandmaster Clock.

11.5.3.1.5 correctionField

The correctionField is a value whose data type is Integer64 and whose units are 2^{-16} ns such that, when combined with the originOrPreciseOriginTimestamp (see 11.5.3.1.4), the result is the value of the Local PTP Clock of the upstream PTP Instance at the arbitrarily selected time X of 11.5.3.1.4.

NOTE 1—See NOTE 1 of 11.5.3.1.4. In Example 1 and Example 2 illustrated by Figure 45, the value of correctionField sent across the MDMI interface at BC-D is the sub-nanosecond portion of the time of X relative to the Local PTP Clock at BC-C, excluding any sub-nanosecond portion.

NOTE 2—See NOTE 2 of 11.5.3.1.4. If Example 2 were modified such that TC-B and TC-E were removed and the other PTP Instances followed the specifications of IEEE Std 802.1AS-2011, then the value of correctionField sent across the MDMI interface at BC-D would be the difference between the time of X relative to the Local PTP Clock at BC-C and the time relative to the Grandmaster Clock that the time synchronization information received at BC-D was sent by the Grandmaster Clock.

11.5.3.1.6 logMessageInterval

The logMessageInterval is the value of the portDS.logSyncInterval member for the upstream PTP Port (i.e., the MASTER port of BC-C in Figure 45) that sent this information. This information is carried by the respective media-specific time transfer mechanism which refers to relevant requirements of IEEE Std 802.1AS-2011 (see 11.5.2.2.3).

11.5.3.1.7 upstreamTxTime

The upstreamTxTime is equal to the value of the Timestamping Clock of this PTP instance at the arbitrarily selected time X of 11.5.3.1.4.

NOTE 1—In Example 1 and Example 2 illustrated by Figure 45, and also in the modification of Example 2 described in 11.5.3.1.4, NOTE 2, X is the time of the syncEventEgressTimestamp of the sending of this time synchronization information by the upstream PTP Instance (i.e., BC-C in Figure 45). In this case, the upstreamTxTime is equal to the <syncEventIngressTimestamp> (see 7.3.3.1), for the receipt of this time synchronization information, minus the mean path delay measured by this PTP Port and corrected for any asymmetry (see 7.4.2). The <syncEventIngressTimestamp> and the mean path delay corrected for any asymmetry are both expressed relative to the Timestamping Clock of this PTP Instance, that is, BC-C. In Figure 45, Example 1, this Timestamping Clock is the Local PTP Clock of BC-D. In Figure 45, Example 2, and in the modification of Example 2 described in 11.5.3.1.4, NOTE 2, this Timestamping Clock is the Local Clock of BC-D.

NOTE 2—The upstreamTxTime and the sum of the originOrPreciseOriginTimestamp (see 11.5.3.1.4) and correctionField (see 11.5.3.1.5) represent the same instant of time.

11.5.3.1.8 cumulativeRateRatio

The cumulativeRateRatio is the measured ratio of the frequency of the Grandmaster Clock to the frequency of the Timestamping Clock of the upstream PTP Instance (see 16.10.3). It is equal to <cumulativeRateRatio> for the upstream PTP instance.

NOTE 1—In Example 1 illustrated in Figure 45, the network does not use the cumulative frequency method, and cumulativeRateRatio is 1. In Figure 45, Example 2, and in the modification of Example 2 described in 11.5.3.1.4, NOTE 2, cumulativeRateRatio is the measured ratio of the frequency of the Grandmaster Clock to the frequency of the Timestamping Clock of BC-C, which is the Local Clock of BC-C in this case.

NOTE 2—Whether the network uses the cumulative frequency method is independent of the media-specific method for transferring frequency across the PTP Communication Path attached to the Special Port. For example, it is possible for the Special Port at each end of a PTP Communication Path to synchronize by measuring the ratio of the frequency of the Special Port at the other end to its frequency, while the network does not use the cumulative frequency method. In this case (and in all cases), the Special Port at the Slave PTP Instance end of the PTP Communication Path is responsible for consistently computing originOrPreciseOriginTimestamp, correctionField, upstreamTxTime, and cumulativeRateRatio of MDSyncSend (see 11.5.3.2).

11.5.3.2 Timing information transfer across the MDMI Interface on transmission

The information transferred across the MDMI Interface, from the PTP Port block to the Special PTP MD Adapter block, is contained in the MDSyncSend structure:

```
MDSyncSend {  
    domainNumber,  
    sdoId,  
    sourcePortIdentity,  
    originOrPreciseOriginTimestamp,  
    correctionField,  
    logMessageInterval,  
    upstreamTxTime,  
    cumulativeRateRatio  
}
```

NOTE 1—In the example of Figure 45, the MDSyncSend information transfer occurs in the egress PTP Port of BC-C.

NOTE 2—In IEEE Std 802.1AS-2011, the member cumulativeRateRatio is renamed “rateRatio.” Both are intended to provide the same information, namely, the value of the cumulativeRateRatio that needs to be distributed to downstream PTP Instances by from the PTP Port in the MASTER state, for example, the Master port of BC-C in the example of Figure 45. For this example, the value is the neighborRateRatio (see 16.10.2), between the BC-C and TC-B times the value of cumulateRateRatio received from TC-B.

The members of the MDSyncSend structure are defined below.

NOTE 3—In the examples, the most recently received time-synchronization information refers to, in this case, the PTP Instance is not the Grandmaster PTP Instance, the information contained in the MDSyncReceive structure that was transmitted across the MDMI Interface on receipt of the most recently received time-synchronization information on the PTP Port in the SLAVE state, for example, BC-C in Figure 45.

11.5.3.2.1 domainNumber

The domainNumber is the number of the PTP domain of this PTP Instance, for example, BC-C in the example of Figure 45.

11.5.3.2.2 sdoId

The sdoId is the sdoId associated with the domainNumber of the PTP domain of this PTP Instance. It is either:

- a) The value specified-by-design, if permitted by 7.1.4, or
- b) The value of the defaultDS.sdoId

of this PTP Instance.

11.5.3.2.3 sourcePortIdentity

The sourcePortIdentity is the portIdentity of this PTP Port.

11.5.3.2.4 originOrPreciseOriginTimestamp

The originOrPreciseOriginTimestamp is a value whose data type is Timestamp (see 5.3.3) such that, when combined with the correctionField (see 11.5.3.2.5), the result is the value of the Local PTP Clock of this PTP Instance at an arbitrarily selected time Y. The time Y represents the same instant of time in 11.5.3.2.4, 11.5.3.2.5, and 11.5.3.2.7, and it is selected by the PTP Port block of this Special Port of the PTP Instance.

NOTE 1—The media-dependent time transfer mechanism includes the measurement of mean path delay and any delay asymmetry. Based on the `originOrPreciseOriginTimestamp`, `correctionField` (see 11.5.3.2.5), `upstreamTxTime` (see 11.5.3.2.7), and `<cumulativeRateRatio>` (see 11.5.3.2.8, note that `<cumulativeRateRatio>` is 1 if the cumulative frequency method is not used) conveyed in the `MDSyncSendStructure`, and the mean path delay, delay asymmetry, and value of the Timestamping Clock at the time Z (see below), the media-dependent time transfer mechanism conveys the value of the Local PTP Clock and the value of the Timestamping Clock, at a time Z, to the downstream PTP Instance (i.e., BC-D in Figure 45), or it conveys equivalent information. For example, the media-dependent mechanism can transfer the value of the Timestamping Clock at the transmitting PTP Instance, along with the measured mean path delay and delay asymmetry, and the downstream PTP Instance (i.e., BC-D) can compute the value of its Timestamping Clock at time Y. Alternatively, the downstream PTP Instance can measure the mean path delay (and be provided with knowledge of any delay asymmetry). Other schemes are also possible. The time Z can be equal to the time Y, or can be another time.

NOTE 2—In Example 1 and Example 2 illustrated by Figure 45, Y can be any convenient time. Several choices that might be convenient are as follows: a) the time the most recently received time synchronization information on the PTP Port in the SLAVE state of this PTP Instance was sent by the upstream PTP Instance (i.e., TC-B), b) the receipt time of the most recently received time synchronization information on the PTP Port in the SLAVE state of this PTP Instance, and c) a time between the receipt time of the most recently received time synchronization information on the PTP Port in the SLAVE state of this PTP Instance and the time the information of the `MDSyncSend` structure is sent. The time Z also can be any convenient time; one choice is the time the time synchronization information is sent by this PTP Instance.

NOTE 3—If Example 2 were modified such that TC-B and TC-E were removed and the other PTP Instances followed the specifications of IEEE Std 802.1AS-2011, then:

- a) The value of `originOrPreciseOriginTimestamp` sent across the MDMI at BC-C would be the value of the `originOrPreciseOriginTimestamp` member of the `MDSyncReceive` structure sent across the MDMI interface on receipt of the most recently-received time synchronization information
- b) Y would be the time that the information was transmitted from the upstream PTP Instance (i.e., BC-A) relative to the Local PTP Clock of the upstream instance (i.e., BC-A), and
- c) Z would be the time that the information is transmitted by this PTP Instance

11.5.3.2.5 correctionField

The `correctionField` is a value whose data type is Integer64 and whose units are 2^{-16} ns such that, when combined with the `originOrPreciseOriginTimestamp` (see 11.5.3.2.4), the result is the value of the Local PTP Clock of this PTP Instance at the arbitrarily selected time Y of 11.5.3.2.4.

NOTE 1—See NOTE 1 of 11.5.3.2.4.

NOTE 2—See NOTE 2 of 11.5.3.2.4.

NOTE 3—If Example 2 were modified such that TC-B and TC-E were removed and the other PTP Instances followed the specifications of IEEE Std 802.1AS-2011, then:

- a) The value of `correctionField` sent across the MDMI interface at BC-C would be the value of `correctionField` member of the `MDSyncReceive` structure sent across the MDMI interface on receipt of the most recently received time synchronization information.
- b) Y would be the time that the information was transmitted from the upstream PTP Instance (i.e., BC-C in Figure 45) relative to the Local PTP Clock of the upstream PTP Instance, and
- c) Z would be the time that the information is transmitted by this PTP Instance.

11.5.3.2.6 logMessageInterval

The `logMessageInterval` is the value of the `portDS.logSyncInterval` for this PTP Port.

11.5.3.2.7 upstreamTxTime

The upstreamTxTime is equal to the value of the Timestamping Clock at the arbitrarily selected time Y of 11.5.3.2.4.

NOTE—The transmitted time synchronization information contains either a) the value of the Local PTP Clock at an arbitrarily selected time Z (selected by the sender, and not necessarily equal to the time Y of 11.5.3.2.4, 11.5.3.2.5, and 11.5.3.2.7) and the value of the Timestamping Clock at time Z, or b) sufficient information for the receiver to compute this information. The originOrPreciseOriginTimestamp (see 11.5.3.2.4), correctionField (see 11.5.3.2.5), upstreamTxTime (see 11.5.3.2.7), and <cumulativeRateRatio>, together with the <syncEventEgressTimestamp> taken when the time synchronization information is transmitted by this PTP Port, the measured mean path delay, and any delay asymmetry provided, are sufficient to compute the information of a) or b). This information will be sent to the downstream PTP Instance (i.e., BC-D in Figure 45). There are various ways in which this can be done, depending on whether the cumulative frequency method is used and on the implementation. For example:

- a) In Example 1 of Figure 45, the cumulative frequency method is not used, and the Timestamping Clock is the same as the Local PTP Clock. The time Y can be any convenient time, and the time Z can be the transmission time of the time synchronization event message at the MASTER PTP Port of the current PTP Instance. The MASTER PTP Port is a Special Port. The SLAVE PTP Port of the current PTP Instance might or might not be a Special Port. The originOrPreciseOriginTimestamp and correctionField of the MDSyncSend structure together reflect the Local PTP Clock at the time Y. The upstreamTxTime is the value of the Timestamping Clock at the time Y. The value of the Local PTP Clock at the time Z is equal to the value of the Local PTP Clock at the time Y (reflected by the originOrPreciseOriginTimestamp at time Y) plus the <syncEventEgressTimestamp> (given by the value of the Timestamping Clock at time Z) minus the value of the Timestamping Clock at time Y. This is equal to the value of the Timestamping Clock at time Z, since the Local PTP Clock and Timestamping Clock are the same.
- b) In Example 2 of Figure 45, for the case where TC-B and TC-E are not present and the other time-aware systems follow IEEE Std 802.1AS-2011, for transports other than IEEE Std 802.3 EPON, the upstreamTxTime is the value of the upstreamTxTime member of the MDSyncReceive structure sent across the MDMI interface on receipt of the most recently received time synchronization information, and the time Y is the time of transmission of this information by the upstream PTP Port (i.e., the MASTER PTP Port of BC-C). The time Z is the time the time synchronization information is transmitted by this PTP Instance on an outgoing (i.e., MASTER) PTP Port, and this information includes the value of the Local PTP Clock at the time of transmission. The downstream receiver (i.e., the SLAVE PTP Port of BC-D in Figure 45) computes upstreamTxTime for its MDSyncReceive structure from the <syncEventIngressTimestamp>, mean propagation path delay measured relative to the Timestamping Clock, and any known delay asymmetry expressed relative to the Timestamping Clock. The receiver then can compute the time relative to the Local PTP Clock using this information and the cumulativeRateRatio (the cumulative frequency method is used).
- c) In IEEE Std 802.1AS-2011, for IEEE 802.3 EPON transport, the ONU and OLT Timestamping Clocks are synchronized at the physical layer. The EPON MPCP layer computes the propagation delay, which is known to both the OLT and the ONU. The OLT chooses a convenient time Y and transmits this value and the corresponding Local PTP time to the ONU, along with any <cumulativeRateRatio> due to transport from a PTP Instance upstream of the OLT (i.e., BC-A in Figure 45). On reception at the ONU, the value Y is the upstreamTxTime, and the Local PTP Time transmitted by the OLT is reflected by the originOrPreciseOriginTimestamp and correctionField. The times Y and Z are the same.

11.5.3.2.8 cumulativeRateRatio

The cumulativeRateRatio is the measured ratio of the frequency of the Grandmaster Clock to the frequency of the Timestamping Clock of this PTP Instance (see 16.10.3). It is equal to <cumulativeRateRatio> for this PTP Instance.

NOTE—If the network does not use the cumulative frequency method, <cumulativeRateRatio> is 1.

11.5.3.3 Computation of <offsetFromMaster> at the PTP Instance whose PTP Port in the SLAVE state is a Special Port

The <offsetFromMaster> value shall be computed by the Slave PTP Instance as follows:

$$\langle \text{offsetFromMaster} \rangle = \langle \text{syncEventIngressTimestamp} \rangle - \text{originOrPreciseOriginTimestamp} - \text{correctionField} - \langle \text{cumulativeRateRatio} \rangle * (\langle \text{syncEventIngressTimestamp} \rangle - \text{upstreamTxTime})$$

where *originOrPreciseOriginTimestamp*, *correctionField*, *upstreamTxTime*, and *<cumulativeRateRatio>* are respective members of the *MDSyncReceive* structure (see 11.5.3.1.4, 11.5.3.1.5, 11.5.3.1.7, and 11.5.3.1.8, respectively).

NOTE—If the cumulative frequency method is not being used, *<cumulativeRateRatio>* = 1. In this case, *<offsetFromMaster>* = *upstreamTxTime* – *originOrPreciseOriginTimestamp* – *correctionField*. This is equal to *<Time on the Slave PTP Instance>* – *<Time on the Master PTP Instance>* (see 11.2) at the selected time *X* (see 11.5.3.1.4, 11.5.3.1.5, and 11.5.3.1.7).

12. Synchronization and syntonization of clocks

12.1 Clock adjustments

Clock adjustments are procedures to enable a PTP Instance to match the timescale and/or rate of its Local Clock and/or its Local PTP Clock to those of the Grandmaster Clock in the domain, using information from upstream PTP Instances, to within the applicable accuracy requirements.

The clock adjustments shall perform synchronization and/or syntonization. Syntonization and synchronization are described in 12.2 and 12.3, respectively. Unless otherwise specified in an applicable PTP profile, Ordinary Clocks and Boundary Clocks shall perform synchronization.

Clock adjustments may be based on a sequence of PTP timing messages. A clock adjustment need not be applied on the receipt of every PTP timing message.

12.2 Syntonization

12.2.1 General specification

Whether and how the Local PTP Clock and/or the Local Clock serving a PTP Instance is physically syntonized may be specified in the applicable PTP Profile, possibly by the specification of options of this standard, for example, the high accuracy option.

NOTE 1—Here, physical syntonization implies that the rate determined by the physical oscillator-counter combination of the clock, as opposed to the rate determined from a mathematical model based on this combination, agrees to a specified uncertainty with the rate of the clock to which it is syntonized. For example, the cumulative frequency transfer option mechanism (see 16.10) is a syntonization mechanism based on a mathematical model, whereas the techniques used in the high accuracy option (see Annex L) are based on physical syntonization of the oscillator-counter combination.

NOTE 2—The operation of the PTP protocol synchronizes and therefore syntonizes, not necessarily via physical syntonization, the Local PTP Clock of Boundary Clocks and Ordinary Clocks to the Local PTP Clock of the Grandmaster PTP Instance of the domain. A Local Clock of a PTP Instance, and normally the only clock in a Transparent Clock, is free running unless specific steps are taken to physically syntonize it.

12.2.2 Syntonization based on Sync messages

A clock in PTP Instance A may be syntonized to a clock in another PTP Instance B as follows.

For a sequence of Sync, and possibly Follow_Up messages, from PTP Instance B, PTP Instance A computes the value for <correctedMasterEventTimestamp> as follows:

- a) Upon receipt of a Sync message, PTP Instance A generates and records a timestamp <syncEventIngressTimestamp> corrected for latency per 7.3.4. It calculates the value of <correctedSyncCorrectionField> per 11.2 to correct for the <delayAsymmetry> (see 7.4.2) of the path connected to the ingress PTP Port. If the <delayAsymmetry> is not known, its value shall be zero in the calculation.
- b) If the twoStepFlag bit of the flagField of the of the Sync message is FALSE indicating that a Follow_Up message will not be received, then the <correctedMasterEventTimestamp> = <originTimestamp> + <meanDelay> + <correctedSyncCorrectionField>.
- c) If the twoStepFlag bit of the flagField of the of the Sync message is TRUE indicating that a Follow_Up message will be received, then the <correctedMasterEventTimestamp> = <preciseOriginTimestamp> + <meanDelay> + <correctedSyncCorrectionField> + correctionField of Follow_Up message

where

- The <originTimestamp> is the value of the originTimestamp field in the received Sync message.
- The <preciseOriginTimestamp> is the value of the preciseOriginTimestamp field in the received Follow_Up message.
- The <meanDelay> shall be:
 - The value of the <meanPathDelay> specified in 11.3 if the PTP Port is configured to use the delay request-response mechanism, or
 - The value of the <meanLinkDelay> specified in 11.4 if the PTP Port is configured to use the peer-to-peer delay mechanism, or
 - Zero (see 8.2.2.4) if the PTP Port is configured not to use either path delay mechanism.

The rate of change of the time of the clock of PTP Instance A is then adjusted using the sequence of <syncEventIngressTimestamp>s and the sequence of <correctedMasterEventTimestamp>s to align it to the rate of change of the time of the clock of PTP Instance B.

NOTE—As one example, the ratio of the frequency of the Local PTP Clock of PTP Instance B to the frequency of the Local PTP Clock of PTP Instance A can be estimated as the ratio of the elapsed time of the Local PTP Clock of PTP Instance A to the elapsed time of the Local PTP Clock of PTP Instance B between a received timestamp and a second received timestamp some number of syncInterval later; as shown in Equation (6).

$$\frac{\langle \text{syncEventEgressTimestamp} \rangle_N - \langle \text{syncEventIngressTimestamp} \rangle_0}{\langle \text{correctedMasterEventTimestamp} \rangle_N - \langle \text{correctedMasterEventTimestamp} \rangle_0} \quad (6)$$

where N is the number of syncInterval separating the timestamps ($N > 0$). The frequency of the Local PTP Clock of PTP Instance A can then be adjusted by this factor.

12.2.3 Syntonization based on other mechanisms

In some networks, there can be physical signals accessible to PTP Instances that can be used to syntonize their clocks. The applicable PTP Profile may specify that such signals are used. A possible use of such signals is described in the high accuracy option of Annex L and the High Accuracy Delay Request-Response Default PTP Profile of I.5.

12.3 Synchronization

If required as specified in 12.1, an Ordinary Clock or Boundary Clock with a PTP Port in the SLAVE state shall synchronize its Local PTP Clock to the Local PTP Clock of its Parent PTP Instance in the synchronization hierarchy established by the best master clock algorithm. While this standard defines a synchronization protocol, the specific means and details of synchronization are out of the scope of this standard. These means of synchronization shall minimize the value `currentDS.offsetFromMaster`, which is computed and stored in the data set by the Slave Clock per 11.2.

13. PTP message formats

13.1 General

The formats and definitions in this clause define how the originator fills in the fields of PTP messages. What the receiver does with them is defined elsewhere in this standard.

In the tables in this clause, the “octets” column indicates the size of the field in octets. The “offset” column indicates the offset of the first octet of the field from the start of the defined fields of the PTP message.

13.2 General PTP message format requirements

All PTP messages shall have a header, body, and suffix. The suffix may have zero length.

Reserved fields shall be transmitted with all bits of the field set to 0.

NOTE—Reserved fields are ignored by the receiver.

Unless otherwise specified, all PTP message field values (i.e., fields specified in Clause 13 and its subclauses) shall be one of the following:

- An inherent characteristic of the PTP message and specified in this clause
- Instantiated by the PTP Instance originating the PTP message, that is, the originating PTP Instance, based on the data sets or protocol operation of the originating PTP Instance

In the case of two-step Transparent Clocks, any generated Follow_Up and Pdelay_Resp_Follow_Up messages shall, except for adjustments required for the operation of the Transparent Clock, appear as though the PTP messages were generated by the originator of the Sync and Pdelay_Resp messages respectively; see Clause 11.

The data type of the field shall be the type indicated in parentheses in the title of each subclause.

13.3 Header

13.3.1 General header specifications

The common header for all PTP messages shall be as specified in Table 35.

Table 35—Common PTP message header

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
majorSdoId				messageType				1	0
minorVersionPTP				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
minorSdoId								1	5
flagField								2	6
correctionField								8	8
messageTypeSpecific								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

13.3.2 Header specifications

13.3.2.1 majorSdoId (Nibble)

Except for PTP management messages, the value for the majorSdoId field shall be:

- The value specified-by-design if permitted by 7.1.4, else
- The value of the most significant 4 bits of defaultDS.sdoId.

For PTP management messages, the value shall be as defined in 15.4.1.1.

13.3.2.2 minorSdoId (UInteger8)

Except for PTP management messages, the value for the minorSdoId field shall be:

- The value specified-by-design if permitted by 7.1.4, else
- The value of the least significant 8 bits of defaultDS.sdoId.

For PTP management messages, the value shall be as defined in 15.4.1.1.

13.3.2.3 messageType (Enumeration4)

The value messageType shall indicate the type of the PTP message as defined in Table 36.

Table 36—Values of messageType field

PTP message type	Message class	Value (hex)
Sync	Event	0
Delay_Req	Event	1
Pdelay_Req	Event	2
Pdelay_Resp	Event	3
Reserved	—	4–7
Follow_Up	General	8
Delay_Resp	General	9
Pdelay_Resp_Follow_Up	General	A
Announce	General	B
Signaling	General	C
Management	General	D
Reserved	—	E–F

The most significant bit of the messageType field divides this field in half between PTP event and general messages.

13.3.2.4 versionPTP (UInteger4)

The value of the versionPTP field shall be the value of portDS.versionNumber member of the originating PTP Instance.

13.3.2.5 minorVersionPTP (UInteger4)

The value of the minorVersionPTP field shall be the value of portDS.minorVersionNumber of the originating PTP Instance.

13.3.2.6 messageLength (UInteger16)

The value of the messageLength shall be the total number of octets that form the PTP message. The counted octets start with the first octet of the PTP common header and include and terminate with the last octet of any suffix or, if there are no suffix members with the last octet of the PTP message as defined in this subclause. Since the messageLength of the PTP messages can vary depending on whether one or more TLVs are appended, the value of messageLength is not a reliable indication of the messageType.

13.3.2.7 domainNumber (UInteger8)

For Ordinary Clocks or Boundary Clocks, the value of domainNumber shall be the value of the defaultDS.domainNumber of the originating Ordinary Clock or Boundary Clock.

For peer-to-peer delay mechanism messages originating from a peer-to-peer Transparent Clock, the value shall be the value defined in 11.4.

For PTP management messages, the value shall be the value defined in 15.4.1.1.

13.3.2.8 flagField (Octet[2])

The value of the bits of the flagField array shall be as defined in Table 37. For PTP message types where the bit is not defined in Table 37, the values shall be FALSE.

Table 37—Values of flagField

Octet	Bit	Message types	Name	Description
0	0	Announce, Sync, Follow_Up, Delay_Resp	alternateMasterFlag	FALSE if the PTP Port of the originator is in the MASTER state. Conditions to set the flag to TRUE are specified in 17.2 and 17.3.
0	1	Sync, Pdelay_Resp	twoStepFlag	For Sync messages, if there is a Follow_Up message associated with the Sync message the twoStepFlag shall be TRUE, otherwise it shall be FALSE. For Pdelay_Resp messages, if there is a Pdelay_Resp_Follow_Up message associated with the Pdelay_Resp message the twoStepFlag shall be TRUE, otherwise it shall be FALSE.
0	2	ALL	unicastFlag	TRUE, if the transport layer protocol address to which this PTP message was sent is a unicast address. FALSE, if the transport layer protocol address to which this PTP message was sent is a multicast address.
0	5	ALL	PTP Profile Specific 1	As defined by the applicable PTP Profile; otherwise FALSE.
0	6	ALL	PTP Profile Specific 2	As defined by the applicable PTP Profile; otherwise FALSE.
0	7			Reserved.
1	0	Announce	leap61	The value of timePropertiesDS.leap61.
1	1	Announce	leap59	The value of timePropertiesDS.leap59.
1	2	Announce	currentUtcOffsetValid	The value of timePropertiesDS.currentUtcOffsetValid.
1	3	Announce	ptpTimescale	The value of timePropertiesDS.ptpTimescale.
1	4	Announce	timeTraceable	The value of timePropertiesDS.timeTraceable.
1	5	Announce	frequencyTraceable	The value of timePropertiesDS.frequencyTraceable.
1	6	Announce	synchronizationUncertain (optional flag)	This is an optional flag. If this option is implemented, the flag shall contain the value of currentDS.synchronizationUncertain.

NOTE 1—The change in the requirement for the twoStepFlag removes the inconsistency between Table 20 and 11.2.5.1 in the 2008 edition and in addition is required by the switch in this edition from the twoStepFlag being a PTP Instance property to a PTP Port property

NOTE 2—Bit 7 of octet 0 is not utilized as an indication for a security extension as it was in the 2008 edition. Whether or not a message is suffixed by an AUTHENTICATION TLV is determined by the applicable security policy while utilizing the policy limiting fields (see 16.14.4.1).

All unassigned flagField bits are reserved.

13.3.2.9 correctionField (Integer64)

The correctionField is the value of the correction measured in nanoseconds and multiplied by 2^{16} . For example, 2.5 ns is represented as 0000000000028000₁₆.

A value of one in all bits, except the most significant, of the field shall indicate that the correction is too big to be represented.

The value of the correctionField depends on the PTP message type as described in Table 38.

Table 38—correctionField semantics

Message type	correctionField description
Sync	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Delay_Req	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Pdelay_Req	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Pdelay_Resp	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Follow_Up	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Delay_Resp	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Pdelay_Resp_Follow_Up	Corrections for fractional nanoseconds, <residenceTime>, <delayAsymmetry> and <meanDelay> (see Clause 10 and Clause 11).
Announce	Zero.
Signaling	Zero.
PTP management	Zero.

13.3.2.10 messageTypeSpecific (Octet[4])

The value of the messageTypeSpecific field shall vary based on the value of the messageType field, as described in Table 39.

Table 39—messageTypeSpecific semantics

Value of messageType	Description
Follow_Up, Delay_Resp, Pdelay_Resp_Follow_Up, Announce, Signaling, Management	For the General message class, this field is reserved (4.2.7, 13.2).
Sync, Delay_Req, Pdelay_Req, Pdelay_Resp	For the Event message class, this field may be used for internal implementation as specified in this subclause.

For PTP event messages only, the four octets of the messageTypeSpecific field may be used for internal implementation of a PTP Instance and its PTP Ports. For example, if the PTP Instance consists of multiple hardware components that are not synchronized, messageTypeSpecific can be used to transfer an internal timestamp between components (e.g., a physical layer chip and the PTP Instance's processor). As a second example, if a Transparent Clock is made up of multiple hardware components that cannot share their timestamps, the messageTypeSpecific field could be used to transfer the arrival timestamp from the Transparent Clock's ingress PTP Port to the Transparent Clock's egress PTP Port.

The messageTypeSpecific field is not used in this standard, and it has no meaning from one PTP Instance to another. In the on-the-wire format at each PTP Port, for all messageType values, the messageTypeSpecific field shall be transmitted with all bits of the field set to 0, and ignored on receipt.

13.3.2.11 sourcePortIdentity (PortIdentity)

The value of the sourcePortIdentity field shall be the value of portDS.portIdentity of the PTP Port that originated this PTP message.

13.3.2.12 sequenceId (UInteger16)

The value of the sequenceId field shall be assigned by the originator of the PTP message in conformance with 7.3.7 except in the case of Follow_Up, Delay_Resp, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages and PTP management messages that are a response to another PTP management message. The sequenceId field values for these exceptions are defined in the references listed in Table 40.

Table 40—References for sequenceId value exceptions

PTP message type	Reference
Follow_Up	9.5.10, Clause 10
Delay_Resp	Clause 11
Pdelay_Resp	Clause 11
Pdelay_Resp_Follow_Up	Clause 11
PTP management	15.4.1.2

13.3.2.13 controlField (UInteger8)

For all transport mappings except IPv4 Annex C, the use of this field is obsolete and the field shall be transmitted with all bits 0 and shall be ignored on receipt.

For the transport mapping of IPv4 Annex C, the use of this field is deprecated except as required by the hardware compatibility option specified in C.4.2. When supporting the hardware compatibility option of C.4.2, the value of controlField depends on the message type defined in the messageType field, and it shall have the value specified in Table 41. When not supporting the hardware compatibility option of C.4, the field shall be transmitted with all bits 0 and shall be ignored on receipt.

NOTE—This field is needed for compatibility with implementations from IEEE Std 1588-2008 that use hardware that was designed to conform to IEEE Std 1588™-2002.

Table 41—controlField enumeration

Message type	controlField value (hex)
Sync	00
Delay_Req	01
Follow_Up	02
Delay_Resp	03
Management	04
All others	05
reserved	06 to FF

13.3.2.14 logMessageInterval (Integer8)

The value of the logMessageInterval field is determined by the type of the PTP message and shall be as defined in Table 42.

Table 42—Values of logMessageInterval field

PTP message type	Value of logMessageInterval
Announce	The value of portDS.logAnnounceInterval in a multicast message. In a unicast PTP message the value should be 7F ₁₆ ; however, it shall be either 7F ₁₆ or the value of portDS.logAnnounceInterval. NOTE—The choice for unicast is to allow existing implementations to conform. However, the actual value is not used in unicast operation.
Sync, Follow_Up	The value of portDS.logSyncInterval in a multicast message, and 7F ₁₆ in a unicast PTP message.
Delay_Resp	The value of portDS.logMinDelayReqInterval in a multicast message, or a unicast message according to 16.9, and 7F ₁₆ in a unicast PTP message negotiated according to 16.1.
Delay_Req	7F ₁₆
Signaling	7F ₁₆
PTP management	7F ₁₆
Pdelay_Req	The value shall be either 7F ₁₆ or a value specified by the applicable PTP Profile.
Pdelay_Resp,	7F ₁₆
Pdelay_Resp_Follow_Up	7F ₁₆

13.4 Suffix

A PTP message is suffixed by a contiguous sequence of zero or more entities of data type TLV. The meaning of the entities is described in Clause 14. The first octet of TLV entity “n+1” shall immediately follow the final octet of TLV entity “n”.

PTP Instances should not append TLVs to PTP event messages unless required by the TLV semantics or the specifications of option 16.13.

Unless otherwise stated in the standard or the applicable PTP Profile, the interpretation of a TLV shall not depend on its position in the PTP message.

If the addition of a TLV would cause the applicable maximum frame size to be exceeded, the TLV shall not be added.

NOTE—Appending TLV entities to a PTP event message is likely to introduce asymmetry in the transmission delay suffered by the PTP messages in passing through non-PTP network devices.

13.5 Announce message

13.5.1 General Announce message specifications

The fields of Announce messages shall be as specified in Table 43.

Table 43—Announce message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
originTimestamp								10	34
currentUtcOffset								2	44
reserved								1	46
grandmasterPriority1								1	47
grandmasterClockQuality								4	48
grandmasterPriority2								1	52
grandmasterIdentity								8	53
stepsRemoved								2	61
timeSource								1	63

13.5.2 Announce message field specifications

13.5.2.1 originTimestamp (Timestamp)

The value of originTimestamp shall be 0 or an estimate no worse than ± 1 s of the PTP Instance Time of the originating PTP Instance when the Announce message was transmitted.

13.5.2.2 currentUtcOffset (Integer16)

The value of currentUtcOffset shall be the value of timePropertiesDS.currentUtcOffset.

13.5.2.3 grandmasterPriority1 (UInteger8)

The value of grandmasterPriority1 shall be the value of parentDS.grandmasterPriority1.

13.5.2.4 grandmasterClockQuality (ClockQuality)

The value of grandmasterClockQuality shall be the value of parentDS.grandmasterClockQuality.

13.5.2.5 grandmasterPriority2 (UInteger8)

The value of grandmasterPriority2 shall be the value of parentDS.grandmasterPriority2.

13.5.2.6 grandmasterIdentity (ClockIdentity)

The value of grandmasterIdentity shall be the value of parentDS.grandmasterIdentity.

13.5.2.7 stepsRemoved (UInteger16)

The value of stepsRemoved shall be the value of currentDS.stepsRemoved of the data set of the PTP Instance issuing this PTP message.

13.5.2.8 timeSource (Enumeration8)

The value of timeSource shall be the value of timePropertiesDS.timeSource.

13.6 Sync and Delay_Req messages

13.6.1 General Sync and Delay_Req message specifications

The fields of Sync and Delay_Req messages shall be as specified in Table 44.

Table 44—Sync and Delay_Req message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
originTimestamp								10	34

13.6.2 Sync and Delay_Req message field specifications

13.6.2.1 originTimestamp (Timestamp)

The value of the originTimestamp field shall be as specified in 9.5.9 and in Clause 10 and Clause 11.

13.7 Follow_Up message

13.7.1 General Follow_Up message specifications

The fields of the Follow_Up message shall be as specified in Table 45.

Table 45—Follow_Up message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
preciseOriginTimestamp								10	34

13.7.2 Follow_Up message field specifications

13.7.2.1 preciseOriginTimestamp (Timestamp)

The value of the preciseOriginTimestamp shall be as specified in 9.5.10 and in Clause 10 and Clause 11.

13.8 Delay_Resp message

13.8.1 General Delay_Resp message specifications

The fields of the Delay_Resp message shall be as specified in Table 46.

Table 46—Delay_Resp message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
receiveTimestamp								10	34
requestingPortIdentity								10	44

13.8.2 Delay_Resp message field specifications

13.8.2.1 receiveTimestamp (Timestamp)

The value of the receiveTimestamp shall be as specified in 9.5.12 and in Clause 10 and Clause 11.

13.8.2.2 requestingPortIdentity (PortIdentity)

The value of the requestingPortIdentity shall be as specified in Clause 10 and Clause 11.

13.9 Pdelay_Req message

13.9.1 General Pdelay_Req message specifications

The fields of the Pdelay_Req message shall be as specified in Table 47.

Table 47—Pdelay_Req message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
originTimestamp								10	34
reserved								10	44

NOTE—The reserved field in the Pdelay_Req message is to make the PTP message length match the length of the Pdelay_Resp message. In some networks and bridges, PTP messages with unequal lengths have different transit times that introduce asymmetry errors.

13.9.2 Pdelay_Req message field specifications

13.9.2.1 originTimestamp (Timestamp)

The value of the originTimestamp shall be as specified in Clause 10 and Clause 11.

13.10 Pdelay_Resp message

13.10.1 General Pdelay_Resp message specifications

The fields of the Pdelay_Resp message shall be as specified in Table 48.

Table 48—Pdelay_Resp message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
requestReceiptTimestamp								10	34
requestingPortIdentity								10	44

13.10.2 Pdelay_Resp message field specifications

13.10.2.1 requestReceiptTimestamp (Timestamp)

The value of the requestReceiptTimestamp shall be as specified in Clause 10 and Clause 11.

13.10.2.2 requestingPortIdentity (PortIdentity)

The value of the requestingPortIdentity shall be as specified in Clause 10 and Clause 11.

13.11 Pdelay_Resp_Follow_Up message

13.11.1 General Pdelay_Resp_Follow_Up message specifications

The fields of the Pdelay_Resp_Follow_Up message shall be as specified in Table 49.

Table 49—Pdelay_Resp_Follow_Up message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
responseOriginTimestamp								10	34
requestingPortIdentity								10	44

13.11.2 Pdelay_Resp_Follow_Up message field specifications

13.11.2.1 responseOriginTimestamp (Timestamp)

The value of the responseOriginTimestamp shall be as specified in Clause 10 and Clause 11.

13.11.2.2 requestingPortIdentity (PortIdentity)

The value of the requestingPortIdentity shall be as specified in Clause 10 and Clause 11.

13.12 Signaling message

13.12.1 Receipt of a Signaling message from another PTP Instance

Based on the indicated fields of a received Signaling message, the PTP message shall be accepted and applied as indicated in Table 50. PTP messages that are not accepted shall be ignored.

Table 50—Acceptance of Signaling messages

targetPortIdentity.clockIdentity equal to	targetPortIdentity.portNumber equal to	TLV specification indicates applicability to:	Apply Action to:
defaultDS.clockIdentity	All 1's	PTP Instance	PTP Instance
		PTP Port	all PTP Ports
	portDS.portNumber	PTP Instance	PTP Instance
		PTP Port	target PTP Port
All 1's	All 1's	PTP Instance	PTP Instance
		PTP Port	all PTP Ports
	portDS.portNumber	PTP Instance	PTP Instance
		PTP Port	target PTP Port

13.12.2 Transmission of a Signaling message

A PTP Port shall issue a Signaling message when required by any of the following:

- A TLV on a received Signaling message
- An optional or mandatory feature of this standard
- Implementation-specific considerations outside the scope of this standard

The Signaling message is used to transport a sequence of one or more TLV entities. Signaling messages are transmitted from one PTP Instance to one or more other PTP Instances.

The common fields of a Signaling message shall be as specified in Table 51.

Table 51—Signaling message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
targetPortIdentity								10	34
One or more TLVs								M	44

13.12.2.1 targetPortIdentity (PortIdentity)

The targetPortIdentity field shall be of type PortIdentity. The value of the targetPortIdentity shall be the portIdentity (see 7.5.2) of the PTP Port to which this PTP message is addressed.

NOTE—See 13.12.1.

13.13 PTP management message

PTP management messages are defined in Clause 15.

14. TLV entity specifications

14.1 General requirements

All TLV extensions shall have the data type, TLV (see 5.3.8).

Throughout Clause 14, the term “supported TLV” means that the PTP Instance implementation executes operations required by specifications (e.g., in an optional feature or in a PTP Profile) applicable to said TLV upon its reception. The term “unsupported TLV” means that the implementation does not execute such operations.

Except for adhering to the propagation specifications (see 14.2), PTP Instances that cannot parse and act on a TLV extension shall ignore it and shall attempt to parse the next TLV in the PTP message. For a Boundary Clock, an ignored TLV attached to an Announce message is handled as specified in 14.2.2.

In the tables in Clause 14, the “octets” column indicates the size of the field in octets. The “TLV offset” column indicates the offset of the first octet of the field from the start of the TLV.

The specifications for TLV contents on transmission and any action on reception are defined in the clauses specifying each TLV as indicated in the “TLV defined in subclause” column of Table 52.

For the operation of a Transparent Clock, refer to 10.1.2.

14.1.1 tlvType (Enumeration16)

The tlvType shall identify the TLV.

The values shall be as specified in Table 52.

Table 52—tlvType values

tlvType values	Value (hex)	TLV defined in subclause	Propagation by Boundary Clock unsupported TLV attached to Announce messages
Reserved	0000	—	Do Not Propagate
MANAGEMENT	0001	15.5.2	Do Not Propagate
MANAGEMENT ERROR STATUS	0002	15.5.2	Do Not Propagate
MANIZATION EXTENSION	0003	14.3	Do Not Propagate
UNICAST TRANSMISSION	0004	16.1	Do Not Propagate
UNICAST TRANSMISSION	0005	16.1	Do Not Propagate
UNICAST TRANSMISSION	0006	16.1	Do Not Propagate
KNOWLEDGE CANCEL UNICAST TRANSMISSION	0007	16.1	Do Not Propagate
TRACE	0008	16.2	Propagate
ERNATE TIME OFFSET INDICATOR	0009	16.3	Propagate
Reserved	000A-1FFF		Do Not Propagate
Reserved for assignment by the IEEE 1588 Working Group for TLVs that do not propagate	2000, 2001, 2002, 2003		Do Not Propagate
Experimental values (see 4.2.9)	2004-202F		Do Not Propagate
Reserved	2030-3FFF		Do Not Propagate
MANIZATION EXTENSION PROPAGATE	4000	14.3	Propagate
ANCED ACCURACY METRICS	4001	16.12	Propagate
Reserved for assignment by the IEEE 1588 Working Group for TLVs that propagate	4002-7FFF		Propagate
Experimental values (see 4.2.9)	7F00-7FFF		Propagate
MANIZATION_EXTENSION_DO_NOT_PROPAGATE	8000	14.3	Do Not Propagate
SYNC	8001	Annex L	Do Not Propagate
T_COMMUNICATION_AVAILABILITY	8002	16.9.2.1	Do Not Propagate
TOCOL_ADDRESS	8003	16.9.2.2	Do Not Propagate
VE_RX_SYNC_TIMING_DATA	8004	16.11.4.1	Do Not Propagate
VE_RX_SYNC_COMPUTED_DATA	8005	16.11.4.2	Do Not Propagate
VE_TX_EVENT_TIMESTAMPS	8006	16.11.5.1	Do Not Propagate
RELATIVE_RATE_RATIO	8007	16.10	Do Not Propagate
	8008	14.4	Do Not Propagate
IDENTIFICATION	8009	16.14	Do Not Propagate
Reserved for assignment by the IEEE 1588 Working Group for TLVs that do not propagate	800A-FFFF		Do Not Propagate
Reserved	FFF0 – FFFF		Do Not Propagate

14.1.2 lengthField (UInteger16)

The value of lengthField is the length of the valueField of the TLV type in octets (see 5.3.8).

14.1.3 valueField (tlvType specific)

The format and meaning of the valueField member of the TLV is defined in subsequent clauses for each tlvType defined in this standard.

14.2 Propagation of TLVs through Boundary Clocks

14.2.1 TLVs Attached to non-Announce PTP Messages

When a TLV:

- is received by a Boundary Clock on a PTP message other than Announce, and
- is supported by the Boundary Clock, and
- is attached to the PTP message specified by the applicable optional feature or applicable PTP Profile,

then the Boundary Clock shall propagate or not propagate the TLV according to the optional feature or applicable PTP Profile. In all other cases of a TLV attached to a PTP message other than Announce, the Boundary Clock shall not propagate the TLV.

For the operation of a Transparent Clock, refer to 10.1.2.

14.2.2 Unsupported TLVs Attached to Announce PTP Message

14.2.2.1 Unsupported TLVs marked “Do Not Propagate”

In some network deployments, it is desirable for a TLV to be blocked from propagating through a Boundary Clock (e.g., from an ingress PTP port in the SLAVE state to an egress PTP port in the MASTER state) when the TLV type is unsupported by the Boundary Clock.

To accommodate such scenarios, the TLV type range from 8000₁₆ to FFEF₁₆ has been defined for nonpropagating TLVs. Also, Table 52 identifies some individual TLVs and additional ranges outside the nonpropagating range as “Do Not Propagate.”

An unsupported TLV within the nonpropagating range, or marked as “Do Not Propagate” according to the Table 52, which is received on an Announce message, shall not be propagated by Boundary Clocks.

For the operation of a Transparent Clock, refer to 10.1.2.

14.2.2.2 Unsupported TLVs marked “Propagate”

In some network deployments, it is desirable for a TLV to be propagated through a Boundary Clock (e.g., from an ingress PTP port in the SLAVE state to an egress PTP port in the MASTER state) even when the TLV is unsupported by the Boundary Clock.

To accommodate such scenarios, the tlvType range from 4000₁₆ to 7FFF₁₆ is defined for propagating TLVs. Additionally, Table 52 identifies some individual TLVs outside the propagating range as “Propagate.”

An unsupported TLV within the propagating range, or marked as “Propagate” according to Table 52, which is received in an Announce message, shall be propagated by a Boundary Clock according to the following specifications:

- Unless otherwise specified in this standard, an instance of such a TLV received in an Announce message shall be attached as soon as possible, but not later than three announceIntervals after receipt on the ingress PTP Port, to a single instance of an Announce message transmitted on each of the other PTP Ports of the receiving Boundary Clock that are transmitting Announce messages, that is, the egress PTP Ports. The transmitted Announce message that contains said TLV should be the first such PTP message transmitted at each egress PTP Port after the transfer of said TLV from the ingress PTP Port to each egress PTP Port via processes internal to the PTP Instance.
- If multiple TLVs are sent to an egress PTP Port from the ingress PTP Port of the PTP Instance prior to transmitting the next Announce message, the TLVs should be appended to the next Announce message in the order of arrival at the egress PTP Port from the ingress PTP Port of the PTP Instance.

NOTE 1—In some architectures, there is a delay in moving information from an ingress to an egress PTP Port.

NOTE 2—In Boundary Clocks where the announceInterval is shorter on an egress PTP Port than it is on the ingress port, there might be several Announce messages transmitted without appended TLVs between those with an appended TLV, even though all Announce messages on the ingress PTP Port have an appended TLV.

For the operation of a Transparent Clock refer to 10.1.2.

14.2.2.3 Supported TLVs received on an Announce message

For the processing of the TLV, refer to the relevant specification in the subclause or applicable PTP Profile that defines the feature using the TLV.

For the operation of a Transparent Clock, refer to 10.1.2.

14.3 Vendor and standard organization extension TLVs

14.3.1 General

Vendor and standard organization extension TLVs can be used by vendors and standard organizations, respectively, to extend the protocol for their specific needs.

14.3.2 TLV member specifications

All organization-specific TLV extensions shall have the format specified in Table 53.

Table 53—Organization-specific TLV fields

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
organizationId								3	4
organizationSubType								3	7
dataField								N	10

14.3.2.1 tlvType (Enumeration16)

The tlvType ORGANIZATION_EXTENSION (see Table 52) is maintained for backward compatibility with existing devices and shall no longer be used for defining extensions on any PTP Messages.

The tlvType shall be ORGANIZATION_EXTENSION for existing extensions defined by vendors and standards organizations.

For extensions conformant to this edition, the following tlvType values must be used.

The tlvType shall be ORGANIZATION_EXTENSION_DO_NOT_PROPAGATE or ORGANIZATION_EXTENSION_PROPAGATE.

14.3.2.2 lengthField (UInteger16)

The value of the lengthField is 6+N, where N is an even number (see 5.3.8).

14.3.2.3 organizationId (Octet[3])

The value shall be the value of an OUI or CID assigned to the vendor or standards organization by the IEEE.¹⁸ The octets shall be assigned, in order, to the 3-octet array with the most significant octet of the OUI or CID assigned to the octet array member with index 0. The organization identified by the OUI or CID shall ensure that organizationSubType fields (see 14.3.2.4) are unique within the scope defined by the organizationId value.

NOTE—The organizationSubType is unique among TLVs of all tlvTypes specified by the organization.

PTP Instances that do not recognize a particular organizationId or organizationSubType shall disregard the contents of the TLV except for the lengthField field.

14.3.2.4 organizationSubType (Enumeration24)

The organizationSubType field defines a subtype within the scope of the organizationId field. The organizationSubType values are assigned by the vendor or standards organization identified by the organizationId.

¹⁸ An OUI or CID can be obtained from the IEEE Registration Authority, <https://standards.ieee.org/regauth>. Tutorials on these assigned numbers can be found on this website.

14.3.2.5 data (organizationSubType and organizationId specific)

The format and meaning of the data field shall be defined by the owner of the pair {organizationId, organizationSubType}.

14.4 PAD TLV (optional)

14.4.1 General

The PAD TLV can be used whenever a PTP Profile specifies an increase of the length of any PTP message beyond the length of the PTP message specified in Clause 13, See 16.13 for an example usage. The minimum length increment is 4 octets which occurs with a pad length of 0 octets.

NOTE—The length of a PTP message can also be extended by use of any TLV defined in Table 52. One option from Table 52 to extend both general and event messages would be the ORGANIZATION_EXTENSION_DO_NOT_PROPAGATE TLV.

14.4.2 PAD TLV specification

The PAD TLV format shall be as specified in Table 54.

Table 54—PAD TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
Pad								N	4

14.4.2.1 tlvType

The value of tlvType shall be PAD.

14.4.2.2 lengthField

The value of the lengthField is N, where N is an even number (see 5.3.8).

14.4.2.3 pad

Each N pad octet shall have the value 00₁₆. The pad octets shall not be used for any purpose other than increasing the length of the PTP message to which the PAD TLV is attached.

15. PTP management messages (optional)

15.1 General

PTP management messages are used to access data set members and to generate certain events defined in this standard.

15.1.1 Selection of management mechanisms

Conformance requirements for the selection of management mechanisms are specified in 8.1.4.3.

15.2 PTP management mechanism

Clause 15 specifies a management mechanism using PTP management messages. All TLVs for the PTP management messages are specified in Clause 15. Management messages may be transmitted as either unicast or multicast messages.

PTP Profiles may specify the use of multicast and/or unicast addressing for PTP management messages (see 7.3.1).

Unless otherwise specified in the applicable PTP Profile, a PTP management message that is a response to another PTP management message may use a different form of addressing from the PTP management message to which it is a response.

NOTE—In IEEE Std 1588-2008, that is, version 2.0, the PTP management messages defined in Clause 15 were designed to manage either a PTP Port or a PTP Node supporting a single PTP domain, that is, what in this edition is called a “PTP Instance.” The terminology has been updated to reflect this distinction. With the exception of the CLOCK_DESCRIPTION message (see 15.5.3.1.2), the PTP management messages of this edition also present in version 2.0 are compatible with version 2.0 devices.

15.3 Processing of PTP management messages

15.3.1 Receipt of a PTP management message

Based on the indicated fields of a received PTP management message, the message shall be accepted and applied as indicated in Table 55. PTP management messages not accepted shall be ignored.

Table 55—Acceptance of PTP management messages

targetPortIdentity.clockIdentity equal to:	targetPortIdentity.portNumber equal to:	managementId applicability (see Table 59)	Apply Action to:
defaultDS.clockIdentity	All 1's	PTP Instance	PTP Instance
		PTP Port	all PTP Ports
	portDS.portNumber	PTP Instance	PTP Instance
		PTP Port	target PTP Port
All 1's	All 1's	PTP Instance	PTP Instance
		PTP Port	all PTP Ports
	portDS.portNumber	PTP Instance	PTP Instance
		PTP Port	target PTP Port

NOTE 1—When a “GET” PTP message is addressed to “all PTP Ports,” the responding PTP Instance sends a separate reply for each PTP Port.

NOTE 2—The DEFAULT_DATA_SET TLV can be used to discover any PTP Instance in the system supporting PTP management messages.

15.3.2 Transmission of a PTP management message

The PTP management message is used to transport a single management TLV entity, that is, MANAGEMENT_TLV or MANAGEMENT_ERROR_STATUS_TLV (see 15.5), from a management node to one or more PTP Instances or from a PTP Instance to a management node.

15.3.3 Boundary Clock retransmission of PTP management messages

A Boundary Clock shall retransmit multicast PTP management messages received on one PTP Port via other PTP Ports, that is, not on the ingress PTP Port, according to the following rules, based on the state of the PTP Ports and the value of the boundaryHops field of the received PTP management message:

- Only multicast PTP management messages received on a PTP Port in the MASTER, SLAVE, UNCALIBRATED, or PRE_MASTER states shall be retransmitted.
- If the received boundaryHops field value is 0, the PTP management message shall not be retransmitted. Otherwise, the Boundary Clock shall decrement the value of the boundaryHops field of the PTP management message by 1 before retransmitting the message.
- If the received boundaryHops field value is greater than 0, the PTP management message shall be retransmitted only via PTP Ports in the MASTER, SLAVE, UNCALIBRATED, or PRE_MASTER states.

15.4 PTP management message format

15.4.1 Common fields

The common fields of a PTP management message shall be as specified in Table 56.

Table 56—PTP management message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 13.3)								34	0
targetPortIdentity								10	34
startingBoundaryHops								1	44
boundaryHops								1	45
reserved			actionField					1	46
reserved								1	47
managementTLV								M	48

15.4.1.1 domainNumber, majorSdoId, and minorSdoId of the header

The domainNumber, majorSdoId, and minorSdoId fields of the PTP message common header (see 13.3) of a PTP management message shall be the same as the domainNumber, majorSdoId, and minorSdoId fields of the PTP Port or PTP Instance on which the PTP management message acts.

15.4.1.2 sequenceId of the header

The sequenceId of the PTP message common header (see 13.3) of a PTP management message with actionField equal to RESPONSE or ACKNOWLEDGE shall be set to the sequenceId of the received PTP management message causing the response. Otherwise the sequenceId shall be as specified in 7.3.7.

15.4.1.3 targetPortIdentity (PortIdentity)

The targetPortIdentity field shall be the portIdentity of the PTP Port on which the PTP management message acts, or it shall be all 1's if the PTP management message acts on all ports of the PTP Instance.

NOTE—The PTP Port identified by targetPortIdentity is not necessarily the PTP Port on which the PTP management message was received.

In the case of a PTP management message transmitted by a PTP Instance to a PTP Management Node, the targetPortIdentity field shall be set to the sourcePortIdentity of the PTP management message to which it is a response.

15.4.1.4 startingBoundaryHops (UInteger8)

The value of the startingBoundaryHops field is implementation specific for PTP messages that are not issued in response to a request from another PTP management message. For PTP management messages that are issued in response to a request from another PTP management message, the value of startingBoundaryHops shall be the value computed from the startingBoundaryHops and boundaryHops fields of the requesting PTP message as (startingBoundaryHops minus boundaryHops).

NOTE—When a PTP management message is received, the absolute value of this difference indicates the number of retransmissions by Boundary Clocks that the PTP message experienced.

15.4.1.5 boundaryHops (UInteger8)

The value of the boundaryHops field indicates the remaining number of successive retransmissions of the PTP management message by Boundary Clocks receiving the PTP message per 15.3.3. The value of boundaryHops shall be identical to the value of the field startingBoundaryHops when first transmitted by the issuing PTP Instance.

15.4.1.6 actionField (Enumeration4)

The value of the actionField shall indicate the action to be taken on receipt of the PTP message as defined in Table 57.

Table 57—Values of the actionField

Action	Action taken	Value (hex)
GET	The PTP management message shall carry a single management TLV. The managementId field of the TLV indicates the specific information that needs to be retrieved. An addressed PTP Port or PTP Instance shall respond with a PTP management message with the actionField value set to RESPONSE. If no error occurs, the TLV in the PTP management message shall contain the current values of the data identified by the managementId. If an error occurs, the TLV shall be a management error status TLV.	0
SET	The PTP management message shall carry a single management TLV. The managementId field of the TLV indicates the specific information values that need to be set. If the data identified by the managementId consists of several fields, the update shall be considered as an atomic action and the failure to update any item shall be considered an error in the execution of the SET. Attempts to set a static or nonconfigurable value shall return a management error status TLV (see 15.5.4). TLVs with data definitions that mix configurable and nonconfigurable data are not permitted. An addressed PTP Port or PTP Instance shall respond with a PTP management message with the actionField value set to RESPONSE. If no error occurs, the TLV in the PTP management message shall contain the current values of the data identified by the managementId. If an error occurs, the TLV shall be a management error status TLV.	1
RESPONSE	The PTP management message shall carry a single management or management error status TLV. The value of the managementId shall be identical to that in the requesting PTP management message. If no error has occurred, the TLV shall be a management TLV containing the current values of the data identified by the managementId field of the PTP management message with the GET or SET actionField. If an error occurs, that is, the action required by the GET or SET message could not be fully executed, the TLV shall be a management error status TLV (see 15.5.4).	2
COMMAND	The PTP management message shall carry a single management TLV. The event indicated by the managementId field shall be initiated. The results of this command shall be acknowledged by a PTP management message with actionField set to ACKNOWLEDGE.	3
ACKNOWLEDGE	The PTP management message shall carry a single management or management error status TLV. If the command is executed correctly, the managementTLV field in the acknowledge PTP message shall be identical to that in the corresponding command PTP management messages and so the managementID and the dataField are identical. If the command could not be executed, the managementTLV field in the acknowledge PTP message shall follow the specifications of 15.5.4 for the MANAGEMENT ERROR STATUS TLV.	4
Reserved		5-F

15.4.1.7 managementTLV

PTP management messages shall be suffixed with a single management or management error status TLV.

15.5 Management TLVs

15.5.1 Management TLV introduction

15.5.1.1 General

Subclause 15.5 details the structure of the MANAGEMENT TLV and the MANAGEMENT_ERROR_STATUS TLV.

There are two forms of MANAGEMENT TLVs: those that manipulate data sets or individual data set members and those that initiate events.

15.5.1.1.1 Management of data sets

PTP-defined configurable attributes, whether maintained in the data sets of Clause 8 or in implementation-specific form, are read and updated by PTP management messages with the actionField value GET and SET, respectively. The TLV data structures for these PTP messages contain only configurable variables.

15.5.1.1.2 Management of events

For TLVs that initiate events, the actionField value of the PTP management message is COMMAND. The event and initiation semantics are defined or referenced for each TLV managementId.

15.5.2 MANAGEMENT TLV field format

MANAGEMENT TLV shall have the format specified in Table 58.

Table 58—MANAGEMENT TLV fields

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
managementId								2	4
dataField								N	6

15.5.2.1 tlvType (Enumeration16)

The tlvType shall be MANAGEMENT.

15.5.2.2 lengthField (UInteger16)

The value of the lengthField is 2+N, where N is an even number (see 5.3.8).

15.5.2.3 managementId (Enumeration16)

The values of the managementId field are defined in Table 59. TLV semantics for each managementId value shall be as defined in subclauses of 15.5 and in Clause 16 and Clause 17.

The entries in the allowed actions column of Table 59 indicate the permissible values of the actionField field in the PTP management message common fields (see 15.4.1.6). The receipt of a PTP management message with a disallowed actionField value shall:

- Cause the contents of the management TLV to be disregarded
- Return a management error status TLV, NOT_SUPPORTED (see 15.5.4)

Table 59 refers to the managed PTP Instance. For specifications for receipt of a management TLV with a RESPONSE or ACKNOWLEDGE actionField, refer to Table 57.

Table 59—managementId values

managementId name	managementId value (hex)	Allowed actions	Applies to
Applicable to all PTP Instance types	0000 – 1FFF		
NULL_PTP_MANAGEMENT	0000	GET, SET, COMMAND	PTP Port
CLOCK_DESCRIPTION	0001	GET	PTP Port
USER_DESCRIPTION	0002	GET, SET	PTP Instance
SAVE_IN_NON_VOLATILE_STORAGE	0003	COMMAND	PTP Instance
RESET_NON_VOLATILE_STORAGE	0004	COMMAND	PTP Instance
INITIALIZE	0005	COMMAND	PTP Instance
FAULT_LOG	0006	GET	PTP Instance
FAULT_LOG_RESET	0007	COMMAND	PTP Instance
Reserved	0008 – 1FFF	—	—
Applicable to Ordinary Clock and Boundary Clocks	2000 – 2FFF	—	—
DEFAULT_DATA_SET	2000	GET	PTP Instance
CURRENT_DATA_SET	2001	GET	PTP Instance
PARENT_DATA_SET	2002	GET	PTP Instance
TIME_PROPERTIES_DATA_SET	2003	GET	PTP Instance
PORT_DATA_SET	2004	GET	PTP Port
PRIORITY1	2005	GET, SET	PTP Instance
PRIORITY2	2006	GET, SET	PTP Instance
DOMAIN	2007	GET, SET	PTP Instance
SLAVE_ONLY	2008	GET, SET	PTP Instance
LOG_ANNOUNCE_INTERVAL	2009	GET, SET	PTP Port
ANNOUNCE_RECEIPT_TIMEOUT	200A	GET, SET	PTP Port
LOG_SYNC_INTERVAL	200B	GET, SET	PTP Port

managementId name	managementId value (hex)	Allowed actions	Applies to
VERSION_NUMBER	200C	GET, SET	PTP Port
ENABLE_PORT	200D	COMMAND	PTP Port
DISABLE_PORT	200E	COMMAND	PTP Port
TIME	200F	GET, SET	PTP Instance
CLOCK_ACCURACY	2010	GET, SET	PTP Instance
UTC_PROPERTIES	2011	GET, SET	PTP Instance
TRACEABILITY_PROPERTIES	2012	GET, SET	PTP Instance
TIMESCALE_PROPERTIES	2013	GET, SET	PTP Instance
UNICAST_NEGOTIATION_ENABLE	2014	GET, SET	PTP Port
PATH_TRACE_LIST	2015	GET	PTP Instance
PATH_TRACE_ENABLE	2016	GET, SET	PTP Instance
GRANDMASTER_CLUSTER_TABLE	2017	GET, SET	PTP Instance
UNICAST_MASTER_TABLE	2018	GET, SET	PTP Port
UNICAST_MASTER_MAX_TABLE_SIZE	2019	GET	PTP Port
ACCEPTABLE_MASTER_TABLE	201A	GET, SET	PTP Instance
ACCEPTABLE_MASTER_TABLE_ENABLED	201B	GET, SET	PTP Port
ACCEPTABLE_MASTER_MAX_TABLE_SIZE	201C	GET	PTP Instance
ALTERNATE_MASTER	201D	GET, SET	PTP Port
ALTERNATE_TIME_OFFSET_ENABLE	201E	GET, SET	PTP Instance
ALTERNATE_TIME_OFFSET_NAME	201F	GET, SET	PTP Instance
ALTERNATE_TIME_OFFSET_MAX_KEY	2020	GET	PTP Instance
ALTERNATE_TIME_OFFSET_PROPERTIES	2021	GET, SET	PTP Instance
Reserved	2022 – 2FFF	—	—
Optional PTP management messages applicable to Ordinary Clock and Boundary Clocks	3000 – 3FFF		
EXTERNAL_PORT_CONFIGURATION_ENABLED	3000	GET, SET (see 17.6)	PTP Instance
MASTER_ONLY	3001	GET, SET	PTP Port
HOLDOVER_UPGRADE_ENABLE	3002	GET, SET	PTP Instance
EXT_PORT_CONFIG_PORT_DATA_SET	3003	GET, SET	PTP Port
Reserved	3004 – 3FFF	—	—
Applicable to Transparent Clocks	4000 to 4FFF	—	—
TRANSPARENT_CLOCK_DEFAULT_DATA_SET	4000	GET	PTP Instance
TRANSPARENT_CLOCK_PORT_DATA_SET	4001	GET	PTP Port
PRIMARY_DOMAIN	4002	GET, SET	PTP Instance
Reserved	4003 – 4FFF	—	—
Optional PTP management messages applicable to Transparent Clocks	5000 – 5FFF		
Reserved	5000 – 5FFF	—	—

Applicable to Ordinary Clocks, Boundary Clocks, and Transparent Clocks	6000 – 7FFF	—	—
DELAY MECHANISM	6000	GET, SET	PTP Port
LOG MIN PDELAY REQ INTERVAL	6001	GET, SET	PTP Port
Reserved	6002 – BFFF	—	—
This range is to be used for implementation-specific identifiers	C000 – DFFF	—	—
This range is to be assigned by an alternate PTP Profile	E000 – FFFE	—	—
Reserved	FFFF	—	—

NOTE—The implementation-specific range of managementIds are assigned by manufacturers to define management functions unique to their own devices. There is no expectation of interoperability, and users need to ensure that such TLVs are directed to the appropriate device.

15.5.3 Management TLV data field specifications for each managementId

15.5.3.1 TLV data fields for MANAGEMENT_TLVs applicable to all PTP Instances

15.5.3.1.1 NULL_PTP_MANAGEMENT

The management TLV data field is of zero length. No action affecting data sets or state shall result from receiving this TLV. The receipt of a NULL_PTP_MANAGEMENT message shall adhere to the requirements of the actionField (see 15.4.1.6).

NOTE—Null PTP management messages are typically used to test implementations by exercising the management handlers without producing any change in protocol operation. For example, such a PTP message can be sent to test whether received PTP management messages are being recorded in an implementation-specific event log.

15.5.3.1.2 CLOCK_DESCRIPTION

The targetPortIdentity.portNumber member of the field (see 15.4.1.3) of the PTP management message carrying this TLV shall indicate the PTP Port to which the physicalLayerProtocol, physicalAddress, protocolAddress, and profileIdentifier apply.

All other fields of this TLV apply to the entire PTP Instance and shall be returned by a query directed at any PTP Port on the PTP Instance.

NOTE—As noted in 15.2, the interpretation and specification of this PTP management message differs between the 2008 edition of this standard and the current edition of this standard. The names of the TLV fields have been revised. With the exception of the clockType field (see 15.5.3.1.2.1), the specification of the values of these fields are compatible. However, for several of the fields, for example, profileIdentifier, the value in the 2008 edition of this standard was static and not part of a data set, while in the current edition of this standard, the value is stored in a data set member. This does not present compatibility issues as the values in the message fields are identical.

The data field shall be as specified in Table 60.

Table 60—CLOCK_DESCRIPTION management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
clockType								2	0
physicalLayerProtocol								L	2
physicalAddressLength								2	2+L
physicalAddress								S	4+L
protocolAddress								N	4+L+S
manufacturerIdentity								3	4+L+S+N
reserved								1	4+L+S+N+3
productDescription								P	4+L+S+N+4
revisionData								Q	4+L+S+N+4+P
userDescription								R	4+L+S+N+4+P+Q
profileIdentifier								6	4+L+S+N+4+P+Q+R
pad								M	4+L+S+N+4+P+Q+R+6

15.5.3.1.2.1 clockType (Boolean[16])

The value clockType shall indicate the type of PTP Instance, as defined in Table 61. A TRUE value of a bit in the clockType field indicates that the description applies to the PTP Instance.

NOTE 1—In both the 2008 edition and the current edition of this standard, the potential clockType field values are identical to those in Table 61. In the current edition, the possible PTP Instance types are enumerated in Table 8 and the applicable value is specified in defaultDS.instanceType. Table 8 values are used for non-PTP management messages, see 8.2.1.5.5. For the CLOCK_DESCRIPTION TLV Table 61 is used so that implementations conformant to the 2008 edition can interpret the field. In Table 42 of the 2008 edition, which corresponds to Table 61 in this edition, more than one bit field can be true, for example, indicating that ordinary clock is combined with an end-to-end transparent clock (this is deprecated in this edition). This circumstance is handled differently in this edition, see NOTE 4—below.

This is a static value not part of the PTP Instance data sets of Clause 8.

NOTE 2—In both the 2008 edition and the current edition of this standard, the value of clockType describes the implementation supporting the PTP domain in which the PTP Port identified communicates. The PTP Port is identified by the targetPortIdentity of the common header.

NOTE 3—In the 2008 edition of this standard, there are five types of PTP device (see 7.6.1 in IEEE Std 1588-2008). In the current edition of this standard, the PTP Management Node is not a PTP Instance.

NOTE 4—In the 2008 edition of this standard, the clockType was interpreted in a note as describing either a single clockType, for example, an Ordinary Clock, or as a device that combined more than a single clockType, for example, an Ordinary Clock and an end-to-end Transparent Clock—a device commonly used in a daisy-chain topology. In the current edition of this standard, this interpretation is deprecated. However, Figure 12 of 6.5.5 illustrates such a device but the two elements, that is, the Transparent Clock and the Ordinary Clock, are distinct PTP Instances and are managed independently. Therefore, only a single bit of the clockType field can be TRUE.

Table 61—clockType specification

Octet	Bit	Description
0	7	Ordinary Clock
0	6	Boundary Clock
0	5	peer-to-peer Transparent Clock
0	4	end-to-end Transparent Clock
0	3	PTP Management Node (deprecated)
0	2,1,0	Reserved
1	all	Reserved

15.5.3.1.2.2 physicalLayerProtocol (PTPText)

The value of physicalLayerProtocol shall indicate the physical layer protocol defining the physicalAddress member. This is a static value not part of the PTP Instance data sets of Clause 8.

The maximum number of symbols in this field shall be 32 (see 5.3.9).

15.5.3.1.2.3 physicalAddressLength (UInteger16)

The value of physicalAddressLength is the number of octets in the physicalAddress field. The range shall be 0 to 16 octets. This is a static value not part of the PTP Instance data sets of Clause 8.

15.5.3.1.2.4 physicalAddress (Octet[physicalAddressLength])

The value of physicalAddress shall be the physical address of the PTP Port indicated by the targetPortIdentity.portNumber member of the field, for example, the MAC address for an IEEE 802.3 end station. If no physical address exists for a specific network technology, the value shall be of zero length.

This is a static value not part of the PTP Instance data sets of Clause 8.

15.5.3.1.2.5 protocolAddress (PortAddress)

The value of protocolAddress shall be the protocol address used as the source address by the network transport protocol for the PTP Port indicated by the targetPortIdentity.portNumber member of the field. The protocolAddress is specified as a data set member in 8.2.18.3.

NOTE—Depending on the rules for the network transport protocol, a PTP Node with more than one PTP Port might have different protocolAddress values for each PTP Port, or they might all be the same.

15.5.3.1.2.6 manufacturerIdentity (Octet[3])

The value of manufacturerIdentity shall provide the unique identity of the manufacturer of the PTP Instance.

The manufacturerIdentity is specified as a data set member in 8.2.5.2.

15.5.3.1.2.7 productDescription (PTPText)

The productDescription field shall indicate the description of the PTP Instance from the manufacturer.

The productDescription is specified as a data set member in 8.2.5.3.

15.5.3.1.2.8 revisionData (PTPText)

The value shall indicate the revisions for components of the PTP Instance.

The revisionData field is specified as data set member productRevision in 8.2.5.4.

15.5.3.1.2.9 userDescription (PTPText)

The userDescription field shall indicate the user-defined description of the PTP Instance.

The userDescription is specified as a data set member in 8.2.5.5.

15.5.3.1.2.10 profileIdentifier (Octet[6])

The value of profileIdentifier shall identify the PTP Profile implemented by the PTP Port indicated by the targetPortIdentity.portNumber member of the PTP message.

The profileIdentifier is specified as a data set member in 8.2.18.2.

15.5.3.1.2.11 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

15.5.3.1.3 USER_DESCRIPTION

The data field shall be as specified in Table 62.

Table 62—USER_DESCRIPTION management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
userDescription								L	0
pad								M	L

15.5.3.1.3.1 userDescription (PTPText)

This TLV is used to configure the value of the userDescription returned by the CLOCK_DESCRIPTION TLV.

The userDescription is specified as a data set member in 8.2.5.5.

15.5.3.1.3.2 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

15.5.3.1.4 SAVE_IN_NON_VOLATILE_STORAGE

The data field is of zero length. The receipt of this TLV shall cause the data set member nonVolatileStorageDS.save (see 8.2.7.3) to be written to the value TRUE.

15.5.3.1.5 RESET_NON_VOLATILE_STORAGE

The data field is of zero length. The receipt of this TLV shall cause the data set member nonVolatileStorageDS.reset (see 8.2.7.2) to be written to the value TRUE.

15.5.3.1.6 INITIALIZE

The data field shall be as specified in Table 63.

Table 63—INITIALIZE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
initializationKey								2	0

15.5.3.1.6.1 initializationKey (Enumeration16)

The value of the initializationKey field of this PTP message shall be as defined in Table 64.

Table 64—INITIALIZATION_KEY enumeration

INITIALIZATION_KEY	Value (hex)	Definition
INITIALIZE_EVENT	0000	In an Ordinary Clock and Boundary Clock, the receipt of an INITIALIZE PTP message with this key shall cause the data set member defaultDS.instanceEnable (see 8.2.1.5.2) to be written to the value TRUE (from FALSE). For Transparent Clocks, this shall cause any implementation-specific PTP Instance initialization procedures to execute.
Reserved	0001–7FFF	Reserved. No action shall occur as a result of receiving an INITIALIZE PTP message with this initializationKey.
Implementation specific	8000–FFFF	The result is implementation specific.

15.5.3.1.7 FAULT_LOG

The data field of the FAULT_LOG TLV shall be as specified in Table 65. The FAULT_LOG TLV returns a list of fault records as specified by the faultLogDS data set (see 8.2.6).

Table 65—FAULT_LOG management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
numberOfFaultRecords								2	0
faultRecord								N	2
pad								M	2+N

15.5.3.1.7.1 numberOfFaultRecords (UInteger16)

The numberOfFaultRecords is specified as a data set member in 8.2.6.2.

15.5.3.1.7.2 faultRecordList (FaultRecord[numberOfFaultRecords])

The faultRecordList is specified as a data set member in 8.2.6.3.

15.5.3.1.7.3 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

15.5.3.1.8 FAULT_LOG_RESET

The TLV carries no data.

The FAULT_LOG_RESET command shall cause faultLogDS.reset (see 8.2.6.4) to be written with the value TRUE.

15.5.3.2 TLV data fields applicable to Ordinary Clocks and Boundary Clocks

15.5.3.2.1 TIME

This TLV may be used to GET or SET the time. Time originates in the Grandmaster PTP Instance and is distributed by PTP to other PTP Instances in the domain.

The data field shall be as specified in Table 66.

Table 66—TIME management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
currentTime								10	0

15.5.3.2.1.1 currentTime (Timestamp)

The value of currentTime shall be that of defaultDS.currentTime (see 8.2.1.5.1).

Although currentTime is a dynamic member of defaultDS, this management protocol specifies explicit behavior for write as described in 8.1.4.4.

15.5.3.2.2 CLOCK_ACCURACY

This TLV may be used to set the accuracy of the target PTP Instance.

NOTE—The accuracy and the time in the Grandmaster PTP Instance is normally determined by interacting with a primary or application-specific time source, for example, GPS, by means outside the scope of this standard. If the time is set in the Grandmaster PTP Instance by means of the TIME TLV, then it is recommended that the accuracy also be set. Since the clockAccuracy attribute is considered in the operation of the best master clock algorithm, the setting of the clockAccuracy attribute in any PTP Instance by means of this TLV can result in a change of Grandmaster PTP Instance the next time the best master clock algorithm is performed.

When sent to a settable Grandmaster PTP Instance, the normal rules for GET and SET apply (see 15.4.1.6).

The data field shall be as specified in Table 67.

Table 67—CLOCK_ACCURACY management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
clockAccuracy								1	0
reserved								1	1

15.5.3.2.2.1 clockAccuracy (Enumeration8)

The value of clockAccuracy shall be the value of the defaultDS.clockQuality.clockAccuracy member of the data set. The value shall be selected from the clockAccuracy enumeration (see Table 5).

Although clockAccuracy is a dynamic member of defaultDS, this management protocol specifies explicit behavior for write as described in 8.1.4.4.

15.5.3.2.3 ENABLE_PORT

The TLV carries no data.

In an Ordinary Clock and Boundary Clock, the receipt of an ENABLE_PORT PTP message shall cause the data set member portDS.portEnable (see 8.2.15.5.1) to be written to the value TRUE.

15.5.3.2.4 DISABLE_PORT

The TLV carries no data.

In an Ordinary Clock and Boundary Clock, the receipt of an DISABLE_PORT PTP message shall cause the data set member portDS.portEnable (see 8.2.15.5.1) to be written to the value FALSE.

15.5.3.2.5 SLAVE_EVENT_MONITORING management TLV data field

The Slave event monitoring option of 16.11 provides a mechanism for transmitting in a TLV data that describe aspects of PTP message receipt and transmission. The SLAVE_EVENT_MONITORING management TLV specified in Table 68 may be used for access to slaveMonitoringPortDS, the data set of the slave event monitoring option (see 16.11.6).

A PTP Instance with a specific “operation X” disabled, upon receiving a SLAVE_EVENT_MONITORING with a value TRUE indicated for “operation X” shall commence the logging or TLV transmission “operation X” where “operation X” is identified by TLV0, TLV1, or TLV2 in Table 68.

A PTP Instance with a specific “operation X” enabled, upon receiving a SLAVE_EVENT_MONITORING TLV with a value FALSE indicated for “operation X” shall cease the logging or TLV transmission “operation X” where “operation X” is identified by TLV0, TLV1, or TLV2 in Table 68.

Table 68—SLAVE_EVENT_MONITORING management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
reserved	reserved	reserved	reserved	reserved	TLV2	TLV1	TLV0	1	0
reserved								1	1
EPRSTT								1	2
EPRSCT								1	3
EPTETT								1	4
TET								1	5

15.5.3.2.5.1 TLV0 (Boolean)

The value of TLV0 shall be the value of index 0 of slaveMonitoringPortDS.slaveEventMonitoringEnable (SLAVE_RX_SYNC_TIMING_DATA generation).

15.5.3.2.5.2 TLV1 (Boolean)

The value of TLV1 shall be the value of index 1 of slaveMonitoringPortDS.slaveEventMonitoringEnable (SLAVE_RX_SYNC_COMPUTED_DATA generation).

15.5.3.2.5.3 TLV2 (Boolean)

The value of TLV2 shall be the value of index 2 of slaveMonitoringPortDS.slaveEventMonitoringEnable (SLAVE_TX_EVENT_TIMESTAMPS generation).

15.5.3.2.5.4 EPRSTT (UInteger8)

The value of EPRSTT shall be the value of slaveMonitoringPortDS.slaveEventMonitoringEventsPerRx SyncTimingTLV.

15.5.3.2.5.5 EPRSCT (UInteger8)

The value of EPRSCT shall be the value of slaveMonitoringPortDS.slaveEventMonitoringEventsPerRx SyncComputedTLV.

15.5.3.2.5.6 EPTETT (UInteger8)

The value of EPTETT shall be the value of slaveMonitoringPortDS.slaveEventMonitoringEventsPerTx EventTimestampsTLV.

15.5.3.2.5.7 TET (UInteger8)

The value of TET shall be the value of slaveMonitoringPortDS.slaveEventMonitoringTxEventType.

15.5.3.3 TLV data fields applicable to the defaultDS data set of Ordinary Clocks and Boundary Clocks

15.5.3.3.1 DEFAULT_DATA_SET

The data field shall be as specified in Table 69.

Table 69—DEFAULT_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	SO	TSC	1	0
reserved								1	1
numberPorts								2	2
priority1								1	4
clockQuality								4	5
priority2								1	9
clockIdentity								8	10
domainNumber								1	18
reserved								1	19

15.5.3.3.1.1 TSC (Boolean)

The value of TSC shall be the value of defaultDS.twoStepFlag.

15.5.3.3.1.2 SO (Boolean)

The value of SO shall be the value of defaultDS.slaveOnly.

15.5.3.3.1.3 numberPorts (UInteger16)

The value of numberPorts shall be the value of the defaultDS.numberPorts.

15.5.3.3.1.4 priority1 (UInteger8)

The value of priority1 shall be the value of defaultDS.priority1.

15.5.3.3.1.5 clockQuality (ClockQuality)

The value of clockQuality shall be the value of defaultDS.clockQuality.

15.5.3.3.1.6 priority2 (UInteger8)

The value of priority2 shall be the value of defaultDS.priority2.

15.5.3.3.1.7 clockIdentity (ClockIdentity)

The value of clockIdentity shall be the value of defaultDS.clockIdentity.

15.5.3.3.1.8 domainNumber (UInteger8)

The value of domainNumber shall be the value of defaultDS.domainNumber.

15.5.3.3.2 PRIORITY1

The data field shall be as specified in Table 70.

Table 70—PRIORITY1 management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
priority1								1	0
reserved								1	1

15.5.3.3.2.1 priority1 (UInteger8)

The value of priority1 shall be the value of defaultDS.priority1.

15.5.3.3.3 PRIORITY2

The data field shall be as specified in Table 71.

Table 71—PRIORITY2 management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
priority2								1	0
reserved								1	1

15.5.3.3.3.1 priority2 (UInteger8)

The value of priority2 shall be the value of defaultDS.priority2.

15.5.3.3.4 DOMAIN

The data field shall be as specified in Table 72.

Table 72—DOMAIN management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
domainNumber								1	0
reserved								1	1

15.5.3.3.4.1 domainNumber (UInteger8)

The value of domainNumber shall be the value of defaultDS.domainNumber.

15.5.3.3.5 SLAVE_ONLY

The data field shall be as specified in Table 73.

Table 73—SLAVE_ONLY management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	SO	1	0
reserved								1	1

15.5.3.3.5.1 SO (Boolean)

The value of SO shall be the value of defaultDS.slaveOnly.

15.5.3.3.6 PATH_TRACE_LIST PTP management message

This PTP management message TLV may be used to retrieve the current pathTraceDS.list member (see 16.2.2.2.1) from an Ordinary Clock or Boundary Clock. The data field shall be as specified in Table 74.

Table 74—PATH_TRACE_LIST management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
pathSequence								8N	0

15.5.3.3.6.1 pathSequence (ClockIdentity[N])

The value of pathSequence is the list of PTP Instance clockIdentity values in the pathTraceDS.list member (see 16.2.2.2.1).

15.5.3.3.7 PATH_TRACE_ENABLE PTP management message

This PTP management message may be used to enable or disable the path trace mechanism. The PATH_TRACE_ENABLE TLV data field shall be as specified in Table 75.

Table 75—PATH_TRACE_ENABLE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	EN	1	0
reserved								1	1

15.5.3.3.7.1 EN (Boolean)

The value of EN provides access to the data set member pathTraceDS.enable (see 16.2.2.3.1).

15.5.3.3.8 ALTERNATE_TIME_OFFSET_ENABLE PTP management message

The ALTERNATE_TIME_OFFSET_ENABLE management TLV allows the indicated alternate timescale to be enabled or disabled in a PTP Instance.

The ALTERNATE_TIME_OFFSET_ENABLE management TLV data format shall be as specified in Table 76.

Table 76—ALTERNATE_TIME_OFFSET_ENABLE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
keyField								1	0
0	0	0	0	0	0	0	EN	1	1

15.5.3.3.8.1 keyField (UInteger8)

The value of keyField shall indicate the alternate timescale enabled or disabled by this TLV entity (see 16.3.4.4.1.1). A value of FF₁₆ shall indicate that all alternate timescales maintained by the Grandmaster PTP Instance are to be enabled or disabled. If the value is not associated with a maintained alternate timescale, the contents shall be disregarded and a MANAGEMENT_ERROR_STATUS TLV shall be returned.

15.5.3.3.8.2 EN (Boolean)

The value of EN provides access to the data set member alternateTimescaleOffsetsDS.list[keyField].enable (see 16.3.4.4.1.2).

15.5.3.3.9 ALTERNATE_TIME_OFFSET_NAME TLV

The ALTERNATE_TIME_OFFSET_NAME management TLV allows a PTP Instance's alternate timescale displayName to be configured.

The ALTERNATE_TIME_OFFSET_NAME management TLV data format shall be as specified in Table 77.

Table 77—ALTERNATE_TIME_OFFSET_NAME management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
keyField								1	0
displayName								L	1
pad								M	1+L

15.5.3.3.9.1 keyField (UInteger8)

The value of keyField shall indicate the alternate timescale updated or queried by this TLV entity (see 16.3.4.4.1.1).

If the value is FF₁₆ or is not associated with any maintained alternate timescale, the TLV shall be ignored and a MANAGEMENT_ERROR_STATUS TLV returned.

15.5.3.3.9.2 displayName (PTPText)

The value of displayName provides access to the data set member alternateTimescaleOffsetsDS.list[keyField].displayName (see 16.3.4.4.1.6).

15.5.3.3.9.3 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

15.5.3.3.10 ALTERNATE_TIME_OFFSET_MAX_KEY management TLV

The ALTERNATE_TIME_OFFSET_MAX_KEY management TLV allows a PTP Management Node to determine the number of alternate timescales maintained.

The ALTERNATE_TIME_OFFSET_MAX_KEY management TLV data format shall be as specified in Table 78.

Table 78—ALTERNATE_TIME_OFFSET_MAX_KEY management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
maxKey								1	0
reserved								1	1

15.5.3.3.10.1 maxKey (UInteger8)

The value of maxKey provides access to the data set member alternateTimescaleOffsetsDS.maxKey (see 16.3.4.3.1).

15.5.3.3.11 ALTERNATE_TIME_OFFSET_PROPERTIES management TLV (optional)

The ALTERNATE_TIME_OFFSET_PROPERTIES management TLV allows a PTP Instance to be

configured with the timescale offset attributes for an alternate timescale.

The ALTERNATE_TIME_OFFSET_PROPERTIES management TLV data format shall be as specified in Table 79.

Table 79—ALTERNATE_TIME_OFFSET_PROPERTIES management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
keyField								1	0
currentOffset								4	1
jumpSeconds								4	5
timeOfNextJump								6	9
reserved								1	15

15.5.3.3.11.1 keyField (UInteger8)

The value of keyField shall indicate the alternate timescale updated or queried by this TLV entity (see 16.3.4.4.1.1).

If the value is FF₁₆ or is not associated with any maintained alternate timescale, the TLV shall be ignored and a MANAGEMENT_ERROR_STATUS TLV returned.

15.5.3.3.11.2 currentOffset (Integer32)

The value of currentOffset shall be the value of the alternateTimescaleOffsetsDS.list[keyField].currentOffset (see 16.3.4.4.1.3).

15.5.3.3.11.3 jumpSeconds (Integer32)

The value of jumpSeconds shall be the value of the alternateTimescaleOffsetDS.list[keyField].jumpSeconds (see 16.3.4.4.1.4).

15.5.3.3.11.4 timeOfNextJump (UInteger48)

The value of timeOfNextJump shall be the value of the alternateTimescaleOffsetsDS.list[keyField].timeOfNextJump (see 16.3.4.4.1.5).

15.5.3.3.12 Configuration of holdoverUpgradeDS

This PTP management message may be used to enable or disable the holdover upgrade mechanism (see 16.4). The HOLDOVER_UPGRADE_ENABLE TLV data field shall be as specified in Table 80.

Table 80—HOLDOVER_UPGRADE_ENABLE TLV management TLV data fields

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	EN	1	0
Reserved								1	1

15.5.3.3.12.1 EN (Boolean)

A value of EN of TRUE shall cause the value of holdoverUpgradeDS.enable (see 16.4.1.1) to be set to TRUE, thus, indicating that the holdover upgrade mechanism is operational. A value of EN of FALSE shall cause the value of defaultDS.holdoverUpgrade to be set to FALSE, thus, indicating that the holdover upgrade mechanism is not operational.

15.5.3.3.13 GRANDMASTER_CLUSTER_TABLE management TLV data field

This PTP message may be used for configuration of the grandmaster cluster mechanism (see 17.2). This TLV provides access to members of the grandmasterClusterDS (see 17.2.3).

The management TLV data field shall be as specified in Table 81.

Table 81—GRANDMASTER_CLUSTER_TABLE management TLV data field

Bits							Octets	TLV data offset
7	6	5	4	3	2	1 0		
logQueryInterval							1	0
actualTableSize							1	1
portAddress							L	2
pad							M	2+L

If the actualTableSize member of the GRANDMASTER_CLUSTER_TABLE TLV is nonzero and no port address of the receiving PTP Instance is contained in the list of portAddress of the TLV, the PTP management message shall be rejected and the grandmasterClusterDS shall not be updated. For this case, the managementErrorId shall be WRONG_VALUE.

If the portAddress array in the TLV cannot be fully stored, the grandmasterClusterDS shall not be altered and the PTP management message shall be rejected. For this case, the managementErrorId shall be WRONG_LENGTH if due to length mismatch, and GENERAL_ERROR for other failures.

Otherwise, upon receipt of a PTP management message with managementId of GRANDMASTER_CLUSTER_TABLE and action field value of SET, the PTP Instance shall replace the current portAddress members of the grandmasterClusterDS with the portAddress of the PTP management message. The member identifying the recipient PTP Instance shall not be entered into the grandmasterClusterDS. If any member fails to update, a MANAGEMENT_ERROR_STATUS TLV shall be returned with the managementErrorId GENERAL_ERROR.

15.5.3.3.13.1 logQueryInterval (Integer8)

The value of logQueryInterval shall be the value of grandmasterClusterDS.logQueryInterval.

15.5.3.3.13.2 actualTableSize (UInteger8)

The value of actualTableSize shall be the number of entries in the portAddress array.

NOTE—The maximum value of actualTableSize is not permitted to exceed the value of grandmasterClusterDS.maxTableSize (see 17.2.3.2.1).

15.5.3.3.13.3 portAddress (PortAddress[actualTableSize])

The value of portAddress shall be the value of the grandmasterClusterDS.portAddress array.

15.5.3.3.13.4 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

15.5.3.3.14 ACCEPTABLE_MASTER_TABLE management TLV data field

This PTP message may be used for configuration of the acceptable master table mechanism (see 17.5).

The ACCEPTABLE_MASTER_TABLE management TLV data field shall be as specified in Table 82.

If this TLV is received with an action value of SET, the update of actualTableSize and list shall be atomic. If either of these values fail to update, a MANAGEMENT_ERROR_STATUS TLV shall be returned.

Table 82—ACCEPTABLE_MASTER_TABLE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
actualTableSize								2	0
list								L	2
pad								M	2+L

15.5.3.3.14.1 actualTableSize (Integer16)

The value of actualTableSize shall be the value of acceptableMasterTableDS.actualTableSize (see 17.5.3.4.1).

15.5.3.3.14.2 list(acceptableMaster[actualTableSize])

The value of list shall be the value of the acceptableMasterTableDS.list.

The data type of each list member is AcceptableMaster (see 17.5.3.2).

NOTE 1—The 2008 edition of this standard identified an acceptable master by its protocol address. The intent was that the source address carried in the transport layer PDU carrying an Announce message could be compared with the protocol address of each entry of the acceptable master table to see if the Announce message is from an acceptable PTP Port in the MASTER state. However, if the Announce message traverses a Transparent Clock (either end-to-end or peer-to-peer), the Transparent Clock will insert the protocol address of its transmitting PTP Port in the source address field when it transmits the Announce message. The reason this occurs is that PTP messages are transmitted from the PTP layer in a Transparent Clock, which is above the transport layer. When the Announce message is received by a downstream Boundary Clock or Ordinary Clock that has the acceptable master table configured and enabled, it cannot be determined if the Boundary Clock or Ordinary Clock PTP Port where the Announce information originated is a member of the acceptable master table, because the source address field of the transport layer PDU that contains the Announce message has been changed.

The 2008 edition of this standard specified that an acceptable master would be identified by protocol address. The identification of an acceptable master by protocol address is deprecated (see 4.2.8) in the current edition of this standard. New PTP Profiles that use the acceptable master table identify acceptable masters by portIdentity and use the AcceptableMaster structure as specified in 17.5.3.2.

NOTE 2—A new PTP Profile has a `profileName` (see 20.3.3) that is different from that of any existing PTP Profile of the same OUI or CID. For a new PTP Profile, the portion of the `profileIdentifier` prior to the `primaryVersion` and `revisionNumber` octets is different from that of any existing profile.

NOTE 3—If a PTP Profile specifies the use of the acceptable master table and permits the use of Transparent Clocks, then acceptable PTP Ports in the MASTER state can be identified by using the `portIdentities`. The transport protocol source address would not be adequate, as a Transparent Clock inserts the address of its transmitting PTP Port into the source address field of the transport protocol layer overhead.

NOTE 4—The only PTP message that carries the `AcceptableMaster` data type is a PTP management message that carries the `ACCEPTABLE_MASTER_TABLE` management TLV. Since the PTP Management Node can determine which `AcceptableMaster` is supported based on the `profileIdentifier`, backward compatibility can be provided for PTP management messages. The manner in which the acceptable master table is maintained in a PTP Instance is not visible externally. It is only necessary that a BC or OC for which `acceptableMasterPortDS.enabled` is `TRUE` be able to determine whether a received PTP message was sent by an acceptable PTP Port in the MASTER state. In practice, this means that an acceptable master table implementation that conforms with the 2008 edition of this standard will inter-operate with an implementation that conforms with the current edition of this standard if the network does not contain any Transparent Clocks (or if the PTP Profile does not allow Transparent Clocks). This also means that the acceptable master table feature of the current edition of this standard is backward compatible with the acceptable master table feature of the 2008 edition of this standard if the network does not contain any Transparent Clocks or the PTP Profile does not allow Transparent Clocks.

15.5.3.3.14.3 pad (Octet[M])

The `pad` field shall be an octet array of length `M`, where `M` is either 1 or 0. If `M` is 1, all bits in the octet shall be 0. The value of `M` shall be such that the requirements of 15.5.2.2 are met.

15.5.3.3.15 ACCEPTABLE_MASTER_MAX_TABLE_SIZE management TLV data field

This PTP message may be used for configuration of the acceptable master table mechanism (see 17.5).

The `ACCEPTABLE_MASTER_MAX_TABLE_SIZE` management TLV data field shall be as specified in Table 83.

Table 83—ACCEPTABLE_MASTER_MAX_TABLE_SIZE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
maxTableSize								2	0

15.5.3.3.15.1 maxTableSize (UInteger16)

The value of `maxTableSize` shall be the value of `acceptableMasterTableDS.maxTableSize` (see 17.5.3.3.1).

15.5.3.4 TLV data fields applicable to the currentDS data set of Ordinary Clocks and Boundary Clocks

15.5.3.4.1 CURRENT_DATA_SET

The data field shall be as specified in Table 84.

Table 84—CURRENT_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
stepsRemoved								2	0
offsetFromMaster								8	2
meanPathDelay								8	10

15.5.3.4.1.1 stepsRemoved (UInteger16)

The value of stepsRemoved shall be the value of currentDS.stepsRemoved.

15.5.3.4.1.2 offsetFromMaster (TimeInterval)

The value of offsetFromMaster shall be the value of currentDS.offsetFromMaster.

15.5.3.4.1.3 meanPathDelay (TimeInterval)

The value of meanPathDelay shall be the value of currentDS.meanDelay.

15.5.3.5 TLV data fields applicable to the parentDS data set of Ordinary Clocks and Boundary Clocks**15.5.3.5.1 PARENT_DATA_SET**

The data field shall be as specified in Table 85.

Table 85—PARENT_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
parentPortIdentity								10	0
0	0	0	0	0	0	0	PS	1	10
reserved								1	11
observedParentOffsetScaledLogVariance								2	12
observedParentClockPhaseChangeRate								4	14
grandmasterPriority1								1	18
grandmasterClockQuality								4	19
grandmasterPriority2								1	23
grandmasterIdentity								8	24

15.5.3.5.1.1 parentPortIdentity (PortIdentity)

The value of parentPortIdentity shall be the value of the parentDS.parentPortIdentity member of the data set.

15.5.3.5.1.2 PS (Boolean)

The value of PS shall be the value of the parentDS.parentStats member of the data set.

15.5.3.5.1.3 observedParentOffsetScaledLogVariance (UInteger16)

The value of observedParentOffsetScaledLogVariance shall be the value of parentDS.observedParentOffsetScaledLogVariance.

15.5.3.5.1.4 observedParentClockPhaseChangeRate (Integer32)

The value of observedParentClockPhaseChangeRate shall be the value of parentDS.observedParentClockPhaseChangeRate.

15.5.3.5.1.5 grandmasterPriority1 (UInteger8)

The value of grandmasterPriority1 shall be the value of parentDS.grandmasterPriority1.

15.5.3.5.1.6 grandmasterClockQuality (ClockQuality)

The value of grandmasterClockQuality shall be the value of parentDS.grandmasterClockQuality.

15.5.3.5.1.7 grandmasterPriority2 (UInteger8)

The value of grandmasterPriority2 shall be the value of parentDS.grandmasterPriority2.

15.5.3.5.1.8 grandmasterIdentity (ClockIdentity)

The value of grandmasterIdentity shall be the value of parentDS.grandmasterIdentity.

15.5.3.6 TLV data fields applicable to the timePropertiesDS data set of Ordinary Clocks and Boundary Clocks

15.5.3.6.1 TIME_PROPERTIES_DATA_SET

The data field shall be as specified in Table 86.

Table 86—TIME_PROPERTIES_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
currentUtcOffset								2	0
0	0	FTRA	TTRA	PTP	UTCv	LI-59	LI-61	1	2
timeSource								1	3

15.5.3.6.1.1 currentUtcOffset (Integer16)

The value of currentUtcOffset shall be the value of timePropertiesDS.currentUtcOffset.

15.5.3.6.1.2 LI-61 (Boolean)

The value of LI-61 shall be the value of timePropertiesDS.leap61.

15.5.3.6.1.3 LI-59 (Boolean)

The value of LI-59 shall be the value of timePropertiesDS.leap59.

15.5.3.6.1.4 UTCV (Boolean)

The value of UTCV shall be the value of timePropertiesDS.currentUtcOffsetValid.

15.5.3.6.1.5 PTP (Boolean)

The value of PTP shall be the value of timePropertiesDS.ptpTimescale.

15.5.3.6.1.6 TTRA (Boolean)

The value of TTRA shall be the value of timePropertiesDS.timeTraceable.

15.5.3.6.1.7 FTTRA (Boolean)

The value of FTTRA shall be the value of timePropertiesDS.frequencyTraceable.

15.5.3.6.1.8 timeSource (Enumeration8)

The value of timeSource shall be the value of timePropertiesDS.timeSource.

15.5.3.6.2 UTC_PROPERTIES

The data field shall be as specified in Table 87.

Table 87—UTC_PROPERTIES management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
currentUtcOffset								2	0
0	0	0	0	0	UTCV	LI-59	LI-61	1	2
reserved								1	3

15.5.3.6.2.1 currentUtcOffset (Integer16)

The value of currentUtcOffset shall be the value of timePropertiesDS.currentUtcOffset.

15.5.3.6.2.2 LI-61 (Boolean)

The value of LI-61 shall be the value of timePropertiesDS.leap61.

15.5.3.6.2.3 LI-59 (Boolean)

The value of LI-59 shall be the value of timePropertiesDS.leap59.

15.5.3.6.2.4 UTCV (Boolean)

The value of UTCV shall be the value of timePropertiesDS.currentUtcOffsetValid.

15.5.3.6.3 TRACEABILITY_PROPERTIES

The data field shall be as specified in Table 88.

Table 88—TRACEABILITY_PROPERTIES management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	FTRA	TTRA	0	0	0	0	1	0
reserved								1	1

15.5.3.6.3.1 TTRA (Boolean)

The value of TTRA shall be the value of timePropertiesDS.timeTraceable.

15.5.3.6.3.2 FTRA (Boolean)

The value of FTRA shall be the value of timePropertiesDS.frequencyTraceable.

15.5.3.6.4 TIMESCALE_PROPERTIES

The data field shall be as specified in Table 89.

Table 89—TIMESCALE_PROPERTIES management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	PTP	0	0	0	1	0
timeSource								1	1

15.5.3.6.4.1 PTP (Boolean)

The value of PTP shall be the value of timePropertiesDS.ptpTimescale.

15.5.3.6.4.2 timeSource (Enumeration8)

The value of timeSource shall be the value of timePropertiesDS.timeSource.

15.5.3.7 TLV data fields applicable to the portDS data set of Ordinary Clocks and Boundary Clocks

15.5.3.7.1 PORT_DATA_SET

The data field shall be as specified in Table 90.

Table 90—PORT_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
portIdentity								10	0
portState								1	10
logMinDelayReqInterval								1	11
meanLinkDelay								8	12
logAnnounceInterval								1	20
announceReceiptTimeout								1	21
logSyncInterval								1	22
delayMechanism								1	23
logMinPdelayReqInterval								1	24
reserved				versionNumber				1	25

15.5.3.7.1.1 portIdentity (PortIdentity)

The value of portIdentity shall be the value of portDS.portIdentity.

15.5.3.7.1.2 portState (Enumeration8)

The value of portState shall be the value of portDS.portState.

15.5.3.7.1.3 logMinDelayReqInterval (Integer8)

The value of logMinDelayReqInterval shall be the value of portDS.logMinDelayReqInterval.

15.5.3.7.1.4 meanLinkDelay (TimeInterval)

The value of meanLinkDelay shall be the value of portDS.meanLinkDelay.

15.5.3.7.1.5 logAnnounceInterval (Integer8)

The value of logAnnounceInterval shall be the value of portDS.logAnnounceInterval.

15.5.3.7.1.6 announceReceiptTimeout (UInteger8)

The value of announceReceiptTimeout shall be the value of portDS.announceReceiptTimeout.

15.5.3.7.1.7 logSyncInterval (Integer8)

The value of logSyncInterval shall be the value of portDS.logSyncInterval.

15.5.3.7.1.8 delayMechanism (Enumeration8)

The value of delayMechanism shall be the value of the value of portDS.delayMechanism.

15.5.3.7.1.9 logMinPdelayReqInterval (Integer8)

The value of logMinPdelayReqInterval shall be the value of the value of portDS.logMinPdelayReqInterval.

15.5.3.7.1.10 versionNumber (UInteger4)

The value of versionNumber shall be the value of portDS.versionNumber.

15.5.3.7.2 LOG_ANNOUNCE_INTERVAL

The data field shall be as specified in Table 91.

Table 91—LOG_ANNOUNCE_INTERVAL management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
logAnnounceInterval								1	0
reserved								1	1

15.5.3.7.2.1 logAnnounceInterval (Integer8)

The value of logAnnounceInterval shall be the value of portDS.logAnnounceInterval.

15.5.3.7.3 ANNOUNCE_RECEIPT_TIMEOUT

The data field shall be as specified in Table 92.

Table 92—ANNOUNCE_RECEIPT_TIMEOUT management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
announceReceiptTimeout								1	0
reserved								1	1

15.5.3.7.3.1 announceReceiptTimeout (UInteger8)

The value of announceReceiptTimeout shall be the value of portDS.announceReceiptTimeout.

15.5.3.7.4 LOG_SYNC_INTERVAL

The data field shall be as specified in Table 93.

Table 93—LOG_SYNC_INTERVAL management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
logSyncInterval								1	0
reserved								1	1

15.5.3.7.4.1 logSyncInterval (Integer8)

The value of logSyncInterval shall be the value of portDS.logSyncInterval.

15.5.3.7.5 DELAY_MECHANISM

The value of the DELAY_MECHANISM data field shall be the value of portDS.delayMechanism as specified in Table 94.

Table 94—DELAY_MECHANISM management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
delayMechanism								1	0
reserved								1	1

15.5.3.7.5.1 delayMechanism (Enumeration8)

The value of delayMechanism shall be the value of portDS.delayMechanism for Ordinary Clock or Boundary Clocks. For Transparent Clocks, the value shall be the value of transparentClockDefaultDS.delayMechanism if implemented. Otherwise, it shall be obtained from the implementation-specific storage of this value.

15.5.3.7.6 LOG_MIN_PDELAY_REQ_INTERVAL

The data field shall be as specified in Table 95.

Table 95—LOG_MIN_PDELAY_REQ_INTERVAL management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
logMinPdelayReqInterval								1	0
reserved								1	1

15.5.3.7.6.1 logMinPdelayReqInterval (Integer8)

The value of logMinPdelayReqInterval shall be the value of portDS.logMinPdelayReqInterval.

15.5.3.7.7 VERSION_NUMBER

The data field shall be as specified in Table 96 (see 7.5.4).

Table 96—VERSION_NUMBER management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
reserved				versionNumber				1	0
reserved								1	1

15.5.3.7.7.1 versionNumber (UInteger4)

The value of versionNumber shall be the value of portDS.versionNumber.

15.5.3.7.8 EXTERNAL_PORT_CONFIGURATION_ENABLED

The data field shall be as specified in Table 97.

Table 97—EXTERNAL_PORT_CONFIGURATION_ENABLED management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	EPC	1	0
Reserved								1	1

15.5.3.7.8.1 EPC

The value of EPC shall be the value of the defaultDS.externalPortConfigurationEnabled of the PTP Instance, as defined by 7.6.2, whose state is being accessed.

15.5.3.7.9 MASTER_ONLY

The data field shall be as specified in Table 98.

Table 98—MASTER_ONLY management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	MO	1	0
Reserved								1	1

15.5.3.7.9.1 MO (Boolean)

The value of MO shall be the value of portDS.masterOnly (see 8.2.15.5.2) member of the PTP Port data set whose state is being accessed.

15.5.3.7.10 UNICAST_NEGOTIATION_ENABLE

This PTP message may be used to enable or disable the unicast negotiation mechanism (see 16.1).

The UNICAST_NEGOTIATION_ENABLE management TLV format shall be as specified in Table 99.

Table 99—UNICAST_NEGOTIATION_ENABLE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	EN	1	0
Reserved								1	1

15.5.3.7.10.1 EN (Boolean)

If the value of EN is TRUE, access to the data set member unicastNegotiationPortDS.enable (see 8.2.19.2) is enabled. If the value of EN is FALSE, access to the data set member enable is disabled.

15.5.3.7.11 ALTERNATE_MASTER management TLV data field

This PTP message may be used for configuration of the alternate master mechanism (see 17.3). This TLV provides access to members of the alternateMasterPortDS (see 17.3.3).

The alternate master attributes in Table 100 may be updated using a PTP management message with managementId ALTERNATE_MASTER.

If the ALTERNATE_MASTER management TLV is received with an action value of SET, the updates of alternateMasterPortDS.numberOfAlternateMasters, alternateMasterPortDS.logAlternateMulticastSync Interval, and alternateMasterPortDS.transmitAlternateMulticastSync shall be atomic. If either update fails to update, a MANAGEMENT_ERROR_STATUS TLV shall be returned.

The ALTERNATE_MASTER management TLV data format shall be as specified in Table 100.

Table 100—ALTERNATE_MASTER management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	S	1	0
logAlternateMulticastSyncInterval								1	1
numberOfAlternateMasters								1	2
reserved								1	3

15.5.3.7.11.1 S (Boolean)

The value of S shall be the value of alternateMasterPortDS.transmitAlternateMulticastSync.

15.5.3.7.11.2 logAlternateMulticastSyncInterval (Integer8)

The value of logAlternateMulticastSyncInterval shall be the value of alternateMasterPortDS.logAlternateMulticastSyncInterval.

15.5.3.7.11.3 numberOfAlternateMasters (UInteger8)

The value of numberOfAlternateMasters shall be the value of alternateMasterPortDS.numberOfAlternateMasters.

15.5.3.7.12 UNICAST_MASTER_TABLE management TLV data field

This PTP message may be used for configuration of the unicast discovery mechanism (see 17.4). This TLV provides access to members of the unicastDiscoveryPortDS (see 17.4.3).

The UNICAST_MASTER_TABLE management TLV data field shall be as specified in Table 101.

If this TLV is received with an action value of SET, the update of unicastDiscoveryPortDS.logQueryInterval, unicastDiscoveryPortDS.actualTableSize, and unicastDiscoveryPortDS.portAddress shall be atomic. If any of these values fail to update, a MANAGEMENT_ERROR_STATUS TLV shall be returned.

Table 101—UNICAST_MASTER_TABLE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
logQueryInterval								1	0
actualTableSize								2	1
portAddress								L	3
pad								M	3+L

15.5.3.7.12.1 logQueryInterval (Integer8)

The value of logQueryInterval shall be the value of unicastDiscoveryPortDS.logQueryInterval.

15.5.3.7.12.2 actualTableSize (UInteger16)

The value of actualTableSize shall be the value of unicastDiscoveryPortDS.actualTableSize.

15.5.3.7.12.3 portAddress(PortAddress[actualTableSize])

The value of portAddress shall be the value of the unicastDiscoveryPortDS.portAddress array.

15.5.3.7.12.4 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

15.5.3.7.13 UNICAST_MASTER_MAX_TABLE_SIZE management TLV data field

This PTP message may be used for configuration of the unicast discovery mechanism (see 17.4). This TLV provides access to the maxTableSize member of the unicastDiscoveryPortDS (see 17.4.3).

The UNICAST_MASTER_MAX_TABLE_SIZE management TLV data field shall be as specified in Table 102.

Table 102—UNICAST_MASTER_MAX_TABLE_SIZE management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
maxTableSize								2	0

15.5.3.7.13.1 maxTableSize (UInteger16)

The value of maxTableSize shall be the value of the unicastDiscoveryPortDS.maxTableSize.

15.5.3.7.14 ACCEPTABLE_MASTER_TABLE_ENABLED management TLV data field

This PTP message may be used for configuration of the acceptable master table mechanism (see 17.5).

The management TLV data field shall be as specified in Table 103.

Table 103—ACCEPTABLE_MASTER_TABLE_ENABLED management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	EN	1	0
reserved								1	1

15.5.3.7.14.1 EN (Boolean)

The value of EN shall be the value of acceptableMasterPortDS.enable.

15.5.3.7.15 EXT_PORT_CONFIG_PORT_DATA_SET management TLV data field

This PTP message may be used for the external configuration option (see 17.6).

The management TLV data field shall be as specified in Table 104.

Table 104—EXT_PORT_CONFIG_PORT_DATA_SET management TV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
0	0	0	0	0	0	0	EN	1	0
desiredState								1	0
reserved								1	0

15.5.3.7.15.1 desiredState (Enumeration8)

The value of desiredState shall be the value of externalPortConfigurationPortDS.desiredState (see 17.6.3).

15.5.3.8 TLV data fields applicable to the defaultDS data set of Transparent Clocks (deprecated)

15.5.3.8.1 TRANSPARENT_CLOCK_DEFAULT_DATA_SET (deprecated)

The data field shall be as specified in Table 105.

Table 105—TRANSPARENT_CLOCK_DEFAULT_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
clockIdentity								8	0
numberPorts								2	8
delayMechanism								1	10
primaryDomain								1	11

15.5.3.8.1.1 clockIdentity (ClockIdentity)

The value of clockIdentity shall be the value of transparentClockDefaultDS.clockIdentity.

15.5.3.8.1.2 numberPorts (UInteger16)

The value of numberPorts shall be the value of transparentClockDefaultDS.numberPorts.

15.5.3.8.1.3 delayMechanism (Enumeration8)

The value of delayMechanism shall be the value of transparentClockDefaultDS.delayMechanism.

15.5.3.8.1.4 primaryDomain (UInteger8)

The value of primaryDomain shall be the value of transparentClockDefaultDS.primaryDomain.

15.5.3.8.2 DELAY_MECHANISM (deprecated)

The same TLV applicable to the portDS data set of Ordinary Clocks and Boundary Clocks shall be used (see 15.5.3.7.5).

15.5.3.8.3 PRIMARY_DOMAIN (deprecated)

The data field shall be as specified in Table 106.

Table 106—PRIMARY_DOMAIN management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
primaryDomain								1	0
reserved								1	1

15.5.3.8.3.1 primaryDomain (UInteger8)

The value of primaryDomain shall be the value of the transparentClockDefaultDS.primaryDomain member of the transparentClockDefaultDS data set, if implemented; otherwise, the value shall be obtained from the implementation-specific storage of this value.

15.5.3.9 TLV data fields applicable to the transparentClockPortDS data set of Transparent Clocks

15.5.3.9.1 TRANSPARENT_CLOCK_PORT_DATA_SET

The data field shall be as specified in Table 107.

Table 107—TRANSPARENT_CLOCK_PORT_DATA_SET management TLV data field

Bits								Octets	TLV data offset
7	6	5	4	3	2	1	0		
portIdentity								10	0
0	0	0	0	0	0	0	FLT	1	10
logMinPdelayReqInterval								1	11
peerMeanLinkDelay								8	12

15.5.3.9.1.1 portIdentity (PortIdentity)

The value of portIdentity shall be the value of transparentClockPortDS.portIdentity.

15.5.3.9.1.2 FLT (Boolean)

The value of FLT shall be the value of transparentClockPortDS.faultyFlag.

15.5.3.9.1.3 logMinPdelayReqInterval (Integer8)

The value of logMinPdelayReqInterval shall be the value of transparentClockPortDS.logMinPdelayReqInterval.

15.5.3.9.1.4 peerMeanLinkDelay (TimeInterval)

The value of peerMeanLinkDelay shall be the value of transparentClockPortDS.meanLinkDelay.

15.5.3.9.2 LOG_MIN_PDELAY_REQ_INTERVAL

The same TLV applicable to the portDS data set of Ordinary Clocks and Boundary Clocks shall be used (see 15.5.3.7.6).

15.5.4 MANAGEMENT_ERROR_STATUS TLV

15.5.4.1 General

This TLV is returned in either response or acknowledge PTP management messages. The MANAGEMENT_ERROR_STATUS TLV format shall be as specified in Table 108.

Table 108—MANAGEMENT_ERROR_STATUS TLV format

Bits								Octets	TLV Offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
managementErrorId								2	4
managementId								2	6
reserved								4	8
displayData								N	12
pad								M	12+N

15.5.4.2 tlvType

The value of tlvType shall be MANAGEMENT_ERROR_STATUS.

15.5.4.3 lengthField

The lengthField shall be $8 + N + M$, where N is the length of the displayData field and M is the length of the pad field.

15.5.4.4 managementErrorId (Enumeration16)

The value of managementErrorId shall be taken from the enumeration defined in Table 109.

Table 109—managementErrorId enumeration

managementErrorId	Specification	Value (hex)
Reserved	—	0000
RESPONSE_TOO_BIG	The requested operation could not fit in a single response message.	0001
NO_SUCH_ID	The managementId is not recognized.	0002
WRONG_LENGTH	The managementId was identified but the length of the data was wrong.	0003
WRONG_VALUE	The managementId and length were correct but one or more values were wrong.	0004
NOT_SETTABLE	Some of the variables in the set command were not updated because they are not configurable.	0005
NOT_SUPPORTED	The requested operation is not supported in this PTP Instance.	0006
UNPOPULATED	The targetPortIdentity of the PTP management message refers to an entity that is not present in the PTP Instance at the time of the request.	0007
Reserved		0008–BFFF
Implementation specific	This range is to be used for implementation-specific errors.	C000–DFFF
PTP Profile defined	This range is to be assigned by an alternate PTP Profile.	E000–FFFF
GENERAL_ERROR	An error occurred that is not covered by other managementErrorId values.	FFFE
Reserved	—	FFFF

15.5.4.5 managementId (Enumeration16)

The values of the managementId field are defined in Table 59. The managementId field shall contain the managementId corresponding to the managementId of the PTP management TLV that was in error.

15.5.4.6 displayData (PTPText)

This is an optional text field to provide a human-readable explanation of the error. If no text is to be provided, this field should be omitted from the TLV. However, it is also acceptable to include the displayData field with displayData.fieldLength set to 0.

The maximum number of symbols in the displayData.textField field (see 5.3.9) shall be 50.

15.5.4.7 pad (Octet[M])

The pad field shall be an octet array of length M, where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 15.5.2.2 are met.

16. General optional features

16.1 Unicast message negotiation (optional)

16.1.1 General unicast negotiation PTP operation specifications

When this option is both implemented (see 6.1) and enabled (see 16.1.2), then the option shall operate as specified in this clause.

The operation of unicast negotiation is conducted through use of TLVs. These TLVs should be attached to a Signaling message.

A PTP Port (the requester) may request, by transmitting a REQUEST_UNICAST_TRANSMISSION TLV entity, that another PTP Port (the grantor) transmit unicast Announce, Sync, Delay_Resp, or Pdelay_Resp messages.

A request for unicast transmissions may be made, granted, acknowledged, and canceled irrespective of PTP Port state, except that these operations shall not occur in any PTP Port of an Ordinary Clock or Boundary Clock in the INITIALIZING, FAULTY, or DISABLED states or in any PTP Port of a Transparent Clock that is in a fault condition. After the unicast transmission grant is issued, the service then starts as specified later in this clause.

NOTE 1—Unless further restrictions on grant are specified elsewhere (i.e., in the applicable PTP Profile), a requester needs to take into account that it can receive a grant from PTP Ports other than the MASTER state (e.g., from a PTP Port in the SLAVE state). A requester can inquire on the PTP Port state of the grantor by using a management mechanism.

Unicast negotiation and service transmission is permitted as specified in 7.3.1.

A PTP Port that has this option enabled, and that receives a REQUEST_UNICAST_TRANSMISSION TLV entity, shall respond with a GRANT_UNICAST_TRANSMISSION TLV entity. The transmitted GRANT_UNICAST_TRANSMISSION TLV entity grants the request or denies the request. The request is denied by specifying a grant of zero duration.

A PTP Port to which a grant has been made (grantee) may inform the grantor that it no longer needs the granted service. It does this by transmitting a CANCEL_UNICAST_TRANSMISSION TLV entity with the maintainRequest flag of the CANCEL_UNICAST_TRANSMISSION TLV set to FALSE. A grantor receiving a CANCEL_UNICAST_TRANSMISSION TLV with the maintainRequest flag of the TLV set to FALSE shall respond with an ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV and may immediately cease to provide the indicated service.

A grantor may inform the grantee that it is no longer able to provide the granted service. It does this by transmitting a CANCEL_UNICAST_TRANSMISSION TLV entity with the maintainGrant flag of the TLV set to FALSE.

A grantee receiving a CANCEL_UNICAST_TRANSMISSION TLV with the maintainGrant flag of the TLV set to FALSE shall respond with an ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV and should immediately cease to use the indicated service. The grantor should continue to provide the granted service until either an ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV has been received or an implementation-specific number of CANCEL_UNICAST_TRANSMISSION TLVs have been transmitted.

A grantor shall not cancel an existing contract as a result of the receipt of a maintainGrant and/or maintainRequest flag set to TRUE.

When the grant is for Announce or Sync messages, the grantor shall transmit the PTP messages such that the arithmetic mean of the intervals between PTP message transmissions is within $\pm 30\%$ of the granted inter-message period. The granted service shall start at the time of the transmission of the grant message and continue for at least the duration of the grant, unless either the grantor or grantee cancels the grant.

When the grant is of Delay_Resp messages, the grantee shall transmit Delay_Req messages such that the arithmetic mean of the intervals between PTP message transmissions is not less than 90% of the inter-message period granted for Delay_Resp messages.

When the grant is of Delay_Resp messages and Delay_Req messages are received from the grantee with a mean inter-message period no smaller than 90% of the granted inter-message period, the grantor shall respond to each received Delay_Req message with a Delay_Resp message. Unless either the grantor or the grantee cancels the grant, this operation shall start at the time of the transmission of the grant message and continue for at least the duration of the grant. If the mean period between reception of Delay_Req messages is less than 90% of the granted inter message period, the grantor may ignore the excess Delay_Req messages.

In addition, a PTP Instance shall transmit Announce, Sync, and Delay_Req messages such that at least 90% of the inter-message intervals are within $\pm 30\%$ of the value of the granted inter-message period. The interval between successive PTP messages should not exceed twice the value of the granted inter-message period.

NOTE 2—In addition to these transmission requirements, Delay_Req messages are also required to comply with the transmission requirements defined in 9.5.11.2. The use of the timing option defined in item c) of 9.5.11.2 is possible with unicast transmission provided that an appropriate distribution compatible with both sets of requirements is specified in a PTP Profile.

NOTE 3—A minimum number of inter-message intervals is necessary to verify that a PTP Instance meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detail discussion of statistical analyses, see Mood et al. [B40] and Papoulis [B43].

For each PTP message type, only one grant is active between a grantor and a grantee in a given direction. The reception of a grant message granting transmissions of a particular messageType cancels any previous grant made with the grantor and grantee in that direction.

If a unicast contract is negotiated for a particular PTP message type between two PTP Ports, then any multicast messages of this type between the two PTP Ports should be ignored.

When a unicast contract is negotiated for transmitting Delay_Resp messages, then the Delay_Req messages associated with the Delay_Resp messages shall also be unicast.

16.1.2 Unicast negotiation enable

The unicast negotiation mechanism can be enabled or disabled by management of the data set member `unicastNegotiationPortDS.enable` (see 8.2.19.2). By default, this mechanism shall be disabled unless otherwise specified in the applicable PTP Profile.

A PTP Instance with the Unicast Negotiation mechanism disabled shall not do the following:

- Respond to a REQUEST_UNICAST_TRANSMISSION TLV entity
- Transmit a REQUEST_UNICAST_TRANSMISSION TLV entity
- Transmit a GRANT_UNICAST_TRANSMISSION TLV entity

A PTP Port with the unicast negotiation mechanism enabled, upon receiving a management write of `unicastNegotiationPortDS.enable` to FALSE, shall cancel all negotiated grants using the CANCEL_UNICAST_TRANSMISSION TLV as defined in 16.1.4.3. Until a CANCEL_UNICAST_TRANSMISSION TLV has been transmitted to all grantees, the PTP Port shall return the value TRUE in the response to management read of the member `unicastNegotiationPortDS.enable`.

16.1.3 Granting PTP Port operations

If the requester issues a new transmission request before the current agreement expires, the grantor should, if resources permit, respond to that request with a grant that is at least as generous as the unexpired portion of the previous grant.

If the granting PTP Port considers the combination of inter-message period and duration to be unreasonable, the PTP Port should reduce the duration of its grant in preference to reducing the rate.

16.1.3.1 Cancel Requests for Inactive Service (Handling Abnormal Contract Cancel Request)

If a `maintainRequest` or `maintainGrant` flag is set to TRUE in a CANCEL_UNICAST_TRANSMISSION TLV for which there is no active contract, the receiver shall take no action on the message and shall respond with the `maintainRequest` or `maintainGrant` flag set to FALSE in an ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV.

NOTE—This scenario represents an abnormal situation as it would not normally be expected to receive a request to cancel a contract when there is no active contract.

16.1.3.2 Acknowledge Responses for Inactive Service (Handling Abnormal Contract Cancel Acknowledgment)

If a `maintainRequest` or `maintainGrant` flag is set to TRUE in an ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV for which there is no active contract, the receiver shall take no action on the message.

NOTE—This scenario represents an abnormal situation as it would not normally be expected to receive a request to maintain a service contract when there is no active contract, based on 16.1.4.4.

16.1.3.3 Acknowledge Responses Opposite of the Cancel Request (Handling Abnormal Contract Cancel Acknowledgment)

If a `maintainRequest` or `maintainGrant` flag is set to `TRUE` in an `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV for which the `maintainRequest` or `maintainGrant` flag, respectively, was set to `FALSE` in a `CANCEL_UNICAST_TRANSMISSION` TLV, the receiver of the `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV shall treat this circumstance the same as not having received the `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV for that `maintainRequest` or `maintainGrant` flag.

If a `maintainRequest` or `maintainGrant` flag is set to `FALSE` in an `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV for which the `maintainRequest` or `maintainGrant` flag, respectively, was set to `TRUE` in a `CANCEL_UNICAST_TRANSMISSION` TLV, the receiver of the `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV shall treat this circumstance the same as having the service canceled by the sender of the `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV for that `maintainRequest` or `maintainGrant` flag.

NOTE 1—Referring to 16.1.4.4, the `maintainRequest` in the `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV corresponds to the `maintainRequest` in the `CANCEL_UNICAST_TRANSMISSION` TLV. Likewise, the `maintainGrant` in the `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` TLV corresponds to the `maintainGrant` in the `CANCEL_UNICAST_TRANSMISSION` TLV.

NOTE 2—These scenarios represent abnormal situations as it would not normally be expected to receive an acknowledgement that is opposite of the contract request, based on 16.1.4.4.

16.1.3.4 Single-Direction or Bidirectional Cancel Requests

A PTP Instance may request Unicast transmissions and also grant them at the same time. Two PTP instances may have active contracts for the same message type in both directions (i.e., each instance has a bidirectional contract). A PTP Instance with a bidirectional contract may request to cancel a single direction in a single `CANCEL_UNICAST_TRANSMISSION` TLV with either the `maintainRequest` or `maintainGrant` flag set to `FALSE`. A PTP Instance with a bidirectional contract may request to cancel both directions in a single `CANCEL_UNICAST_TRANSMISSION` TLV with both the `maintainRequest` and `maintainGrant` flag set to `FALSE`.

A PTP Instance with a bidirectional contract that receives a `CANCEL_UNICAST_TRANSMISSION` TLV for one way of the service shall respond with a single `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` for one-way of the service. A PTP Instance with a bidirectional contract that receives a single `CANCEL_UNICAST_TRANSMISSION` TLV for both directions of the service shall respond with a single `ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION` for both directions of the service.

16.1.3.5 Multiple TLVs attached to same PTP message

A PTP Instance wishing to cancel multiple services (i.e., Grant and Request) for the same messageType may do any one of the following:

- Issue a single `CANCEL_UNICAST_TRANSMISSION` TLV with both flags set to `FALSE`
- Issue separate `CANCEL_UNICAST_TRANSMISSION` TLVs for each service (with appropriate flags set), with each TLV attached to its own PTP message
- Issue a PTP message with multiple `CANCEL_UNICAST_TRANSMISSION` TLVs attached

The two flags in the CANCEL_UNICAST_TRANSMISSION TLV are “R” for maintaining a request and “G” for maintaining a grant (see 16.1.4.3.4 and 16.1.4.3.5, respectively).

A PTP Instance compliant with this standard should not issue conflicting flags in multiple CANCEL_UNICAST_TRANSMISSION TLVs attached to a PTP message. (For example, issuing a G=FALSE, R=TRUE, followed by a G=TRUE, R=FALSE, would be conflicting and will result in both directions being canceled.)

A PTP Instance receiving a PTP message containing multiple CANCEL_UNICAST_TRANSMISSION TLVs shall handle each request individually in the order in which they were attached and send the ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV as a PTP message with multiple TLVs attached in the same order.

In the case of a PTP message with multiple TLVs attached where the flags in a CANCEL_UNICAST_TRANSMISSION TLV conflicts with an already processed TLV, the response to the subsequent TLV will follow the behavior outlined in the previous sections (for inactive services and conflicting ACK flags), that is, as if they were received separately. Operationally, this results in the flags of multiple TLVs being AND’ed. (For example, issuing a G=FALSE, R=TRUE, followed by a G=TRUE, and R=FALSE, would result in both directions being canceled.)

16.1.4 Unicast TLVs

16.1.4.1 REQUEST_UNICAST_TRANSMISSION TLV specification

The REQUEST_UNICAST_TRANSMISSION TLV format shall be as specified in Table 110.

Table 110—REQUEST_UNICAST_TRANSMISSION TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
messageType				reserved				1	4
logInterMessagePeriod								1	5
durationField								4	6

16.1.4.1.1 tlvType

The value of tlvType shall be REQUEST_UNICAST_TRANSMISSION.

16.1.4.1.2 lengthField

The value of the lengthField is 6.

16.1.4.1.3 messageType (Enumeration4)

The value of messageType shall indicate the PTP message type for the unicast PTP message transmission requested. The coding of the enumeration is identical to that used in the messageType field of PTP message headers (see Table 36 and 13.3.2.3). Requests for unicast PTP messages other than Announce, Sync, Delay_Resp, or Pdelay_Resp messages shall be denied. If unicast transmission is granted for Sync or Pdelay_Resp messages by a two-step PTP Port, then unicast transmission shall also be used for the corresponding Follow_Up and Pdelay_Resp_Follow_Up messages.

16.1.4.1.4 logInterMessagePeriod (Integer8)

The value of logInterMessagePeriod shall be the logarithm, to base 2, of the requested mean period, in seconds, between the requested unicast PTP messages.

16.1.4.1.5 durationField (UInteger32)

The value of durationField shall be the requested number of seconds for which the requested PTP messages shall be transmitted.

16.1.4.2 GRANT_UNICAST_TRANSMISSION TLV specification

The GRANT_UNICAST_TRANSMISSION TLV format shall be as specified in Table 111.

Table 111—GRANT_UNICAST_TRANSMISSION TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
messageType				reserved				1	4
logInterMessagePeriod								1	5
durationField								4	6
Reserved								1	10
0	0	0	0	0	0	0	R	1	11

16.1.4.2.1 tlvType

The value of tlvType shall be GRANT_UNICAST_TRANSMISSION.

16.1.4.2.2 lengthField

The value of the lengthField is 8.

16.1.4.2.3 messageType (Enumeration4)

The value of messageType shall indicate the PTP message type for the unicast PTP message transmission granted. The coding of the enumeration is identical to that used in the messageType field of PTP message headers (see Table 36 and 13.3.2.3). The value shall be identical to the messageType field of the REQUEST_UNICAST_TRANSMISSION TLV request.

16.1.4.2.4 logInterMessagePeriod (Integer8)

The value of logInterMessagePeriod shall be the logarithm, to base 2, of the granted mean period, in seconds, between the requested unicast PTP messages.

16.1.4.2.5 durationField (UInteger32)

The value of durationField shall be the number of seconds for which the PTP messages shall be transmitted. A value of zero shall indicate that the request has been denied.

16.1.4.2.6 R (Renewal Invited) (Boolean)

The value of R shall be TRUE when the granting PTP Port considers that the grant is likely to be renewed if and when the requesting PTP Port repeats its request; otherwise R shall be FALSE.

16.1.4.3 CANCEL_UNICAST_TRANSMISSION TLV specification

The CANCEL_UNICAST_TRANSMISSION TLV format shall be as specified in Table 112.

Table 112—CANCEL_UNICAST_TRANSMISSION TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
messageType				0	0	G	R	1	4
Reserved								1	5

16.1.4.3.1 tlvType

The value of tlvType shall be CANCEL_UNICAST_TRANSMISSION.

16.1.4.3.2 lengthField

The value of the lengthField is 2.

16.1.4.3.3 messageType (Enumeration4)

The value of messageType shall indicate the type of unicast PTP message transmission to be canceled. The coding of the enumeration is identical to that used in the messageType field of PTP message headers (see Table 36, 13.3.2.3).

16.1.4.3.4 R (maintainRequest) (Boolean)

The value of R shall be FALSE when a PTP Port issues the TLV and wishes to inform a grantor that, either it no longer requires the service for the specified messageType, or no request currently exists for the specified messageType.

The value of R shall be TRUE when a PTP Port issues the TLV and wishes to inform a grantor that it still requires the requested service for the specified messageType.

See 16.1.1 for details.

16.1.4.3.5 G (maintainGrant) (Boolean)

The value of G shall be FALSE when a PTP Port issues the TLV and wishes to inform a grantee that either it will no longer provide the service for the specified messageType or no request currently exists for the specified messageType.

The value of G shall be TRUE when a PTP Port issues the TLV and wishes to inform a grantee that it will continue to provide the granted service for the specified messageType.

See 16.1.1 for details.

16.1.4.4 ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV specification

The ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV format shall be as specified in Table 113.

Table 113—ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
messageType				0	0	G	R	1	4
reserved								1	5

16.1.4.4.1 tlvType

The value of tlvType shall be ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION.

16.1.4.4.2 lengthField

The value of the lengthField is 2.

16.1.4.4.3 messageType (Enumeration4)

The value of messageType shall indicate the type of unicast PTP message cancellation transmission being acknowledged. The value shall be identical to the messageType field in the CANCEL_UNICAST_TRANSMISSION TLV to which this PTP message is the acknowledgment.

16.1.4.4.4 R (maintainRequest) (Boolean)

The value of R shall be identical to the R flag in the CANCEL_UNICAST_TRANSMISSION TLV, to which this PTP message is the acknowledgment, except as stated in 16.1.3.1 covering abnormal values received in a CANCEL_UNICAST_TRANSMISSION TLV.

See 16.1.4.3 for details.

16.1.4.4.5 G (maintainGrant) (Boolean)

The value of G shall be identical to the G flag in the CANCEL_UNICAST_TRANSMISSION TLV to which this PTP message is the acknowledgment, except as stated in 16.1.3.1 covering abnormal values received in a CANCEL_UNICAST_TRANSMISSION TLV.

See 16.1.4.3 for details.

16.2 Path trace (optional)

16.2.1 General

When this option is both implemented (see 6.1), and enabled (see 16.2.2.3.1), then the option shall operate as specified in this clause. Subclause 16.2 specifies a mechanism, using a PATH_TRACE TLV, for tracing the route of a PTP Announce message through the PTP Network. The mechanism shall be implemented in Boundary Clocks and in non-slaveOnly Ordinary Clocks. For proper operation, the mechanism must be operational in all such PTP Instances in the domain.

Upon receipt of a PTP Announce message, a Boundary Clock scans the pathSequence member of the PATH_TRACE TLV to see whether its own clockIdentity, that is, the value of defaultDS.clockIdentity, is present, which would indicate that a loop is present. The Boundary Clock appends its clockIdentity to the tail of the pathSequence member of the TLV, and it appends the TLV to outgoing Announce messages.

One of the principal uses of this mechanism is to detect Announce messages endlessly circulating in loops of Boundary Clocks, that is, so-called “rogue” Announce messages. If such a loop is detected, the received Announce message shall be discarded. Such loops are eliminated by spanning tree protocols executing on the underlying network; see 6.2.

NOTE—The mechanism of 9.3.2.5 provides a safeguard against rogue Announce messages introduced by failure or transients in the operation of spanning tree protocols.

In general, a Boundary Clock can receive Announce messages from multiple sources: the current Parent PTP Instance and any number of foreign masters. The pathTraceDS.list (16.2.2.2.1) should be maintained only for Announce messages from the current Parent PTP Instance. Application of this mechanism to PTP messages other than Announce messages from the current Parent PTP Instance is outside the scope of this standard.

16.2.2 pathTraceDS data set specifications

16.2.2.1 General

If the optional pathTraceDS data set is supported, the name and operational conformance, as defined in 8.1.6, of each pathTraceDS member is summarized in Table 114.

Table 114—pathTraceDS operational conformance

Name	OC	BC	E2E TC	P2P TC
list	required	required	N/A	N/A
enable	required	required	N/A	N/A

This data set is contained within the data sets of the PTP Instance (see 8.2.8).

16.2.2.2 Dynamic members of the pathTraceDS data set

16.2.2.2.1 pathTraceDS.list

The PTP Instance shall maintain pathTraceDS.list, an array of ClockIdentity values. The initialization value shall be the empty list.

The pathTraceDS.list shall be initialized to the empty list whenever the PTP Instance updates data sets based on decision code M1 or M2 (see 9.3.5).

If management read is supported for pathTraceDS.list, this data set member returns the list of ClockIdentity values. Management write of pathTraceDS.list shall not be supported.

16.2.2.3 Configurable members of the pathTraceDS data set

16.2.2.3.1 pathTraceDS.enable

The pathTraceDS.enable member allows for enable/disable of the path trace mechanism using management. The data type of pathTraceDS.enable shall be Boolean.

If pathTraceDS.enable is TRUE, the path trace mechanism shall be operational. If pathTraceDS.enable is FALSE, the path trace mechanism shall be inactive. By default, pathTraceDS.enable shall be FALSE unless otherwise specified in the applicable PTP Profile.

16.2.3 Receipt of an Announce message

The following additional specifications shall apply to the processing of Announce messages received from the current parent PTP Instance (see 9.5.3). A PTP Port of a Boundary Clock receiving an Announce message from the current parent PTP Instance shall:

- Scan the pathSequence member of any PATH_TRACE TLV present for a value of the clockIdentity field equal to the value of the defaultDS.clockIdentity member of the receiving PTP Instance, that is, there is a “match.”

- b) Discard the message if the TLV is present and a match is found.
- c) Copy the pathSequence member of the TLV to the pathTraceDS.list member (see 16.2.2.2.1) if the TLV is present and no match is found.

16.2.4 Transmission of an Announce message

The following additional specifications shall apply to the transmission of Announce messages (see 9.5.8).

A PTP Port sending an Announce message shall append a PATH_TRACE TLV to the message. The value of the data field of the PATH_TRACE TLV shall be the pathTraceDS.list member (see 16.2.2.2.1) with the PTP Instance's clockIdentity appended to the tail of the list. If the size of the resulting frame containing the Announce message exceeds the maximum frame size permitted by the network technology in use, the PATH_TRACE TLV shall not be appended.

16.2.5 PATH_TRACE TLV specification

The PATH_TRACE TLV format shall be as specified in Table 115.

Table 115—PATH_TRACE TLV format

Bits							Octets	TLV offset
7	6	5	4	3	2	1		
tlvType							2	0
lengthField							2	2
pathSequence							8N	4

16.2.5.1 tlvType

The value of tlvType shall be PATH_TRACE.

16.2.5.2 lengthField

The value of the lengthField is 8N.

16.2.5.3 pathSequence (ClockIdentity[N])

The value of pathSequence is a list of clock identities.

16.3 Alternate timescale offsets (optional)

16.3.1 General

When this option is both implemented (see 6.1), and enabled (see 16.3.4.4.1.2), then the option shall operate as specified in this clause.

The transmission of an ALTERNATE_TIME_OFFSET_INDICATOR TLV entity from the Grandmaster PTP Instance indicates the offset of an alternate timescale from the timescale in use in the domain, which is either PTP or ARB as specified in 7.2.1. The time measured relative to the timescale PTP or the timescale

ARB maintained by the Local PTP Clock of a PTP Instance is its PTP Instance Time (see 3.1.54).

This option is limited to alternate timescales for which the duration of the second is the same as the duration of the second for the timescale in use for the domain. In addition, this option is limited to alternate timescales whose times differ from the timescale in use for the domain by an integral number of seconds, and in which any discontinuities in the alternate timescale occur exactly at the start of a second in PTP Instance Time.

The mechanism for maintaining and using an alternate timescale shall not change the PTP Instance Time. This mechanism may be defined in a PTP Profile but is otherwise outside of the scope of this standard.

Multiple alternate timescales may be maintained, each communicated with a separate `ALTERNATE_TIME_OFFSET_INDICATOR` TLV. PTP Instances that are designed to support this option when they are the Grandmaster PTP Instance shall maintain resources to support an implementation-specific number of alternate timescales. Each supported alternate timescale shall be identified by the value of the `keyField` field and shall be described by the `displayName` field in the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV. The key values shall be consecutive beginning with 0 and ending with the value of `alternateTimescaleOffsetsDS.maxKey`. Each supported alternate timescale may be enabled or disabled based on the configuration of the `enable` member of the `AlternateTimescale` structure defining each alternate timescale (see 16.3.4.4.1.2).

If for any reason the Grandmaster PTP Instance originating the TLV cannot ensure that the TLV information is valid, the Grandmaster PTP Instance shall cease transmitting the TLV. A Boundary Clock receiving an Announce message from its Parent PTP Instance without the TLV shall not generate a TLV based on the information contained in previous TLVs.

NOTE 1—There is no requirement that the same `keyField` or `displayName` values transmitted by PTP Instances from different manufacturers correspond to the same alternate timescale.

NOTE 2—The alternate timescale mechanism only deals with a granularity of one second.

NOTE 3—When originating the TLV, the Grandmaster PTP Instance needs to consider the latency between itself and the most remote PTP Instance in computing when to transmit the value of the `timeOfNextJump` filed in the TLV.

A PTP Instance shall include an `ALTERNATE_TIME_OFFSET_INDICATOR` TLV entity for each enabled alternate timescale in all Announce messages it transmits, subject to the requirements of 13.4. If an alternate timescale is disabled, the PTP Instance shall not include the corresponding `ALTERNATE_TIME_OFFSET_INDICATOR` TLV entity.

16.3.1.1 Using the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV when the timescale is PTP

When the timescale is PTP, a Grandmaster PTP Instance may enable another PTP Instance to compute a timescale other than UTC, for example, Pacific Standard Time, or an applications-specific timescale by using the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV. If permitted by the specification of 7.2.4, this option may also be used to enable the computation of UTC.

In computing the TLV values `currentOffset`, `jumpSeconds`, and `timeOfNextJump`, the Grandmaster PTP Instance shall be responsible for including all alternate timescale offset information including, if applicable, the value of `<dLS>` and any pending leap seconds. PTP Instances receiving the TLV shall not use the values of `timePropertiesDS.currentUtcOffset`, `timePropertiesDS.leap59`, `timePropertiesDS.leap61`, or of the `currentUtcOffset` or `leap59` and `leap61` flags of a received Announce message in computing time in the alternate timescale.

The `timeOfNextJump`, `jumpSeconds`, and `currentOffset` shall be interpreted relative to the timescale PTP.

The PTP Instance receiving the TLV shall add the TLV's currentOffset to the PTP Instance Time to compute a representation of time in the alternate timescale. The timeOfNextJump and jumpSeconds are used to ensure that discontinuities occur at the correct PTP Instance Time.

16.3.1.2 Using the ALTERNATE_TIME_OFFSET_INDICATOR TLV when the timescale is ARB

When the timescale is ARB, a Grandmaster PTP Instance may enable another PTP Instance to compute an alternate timescale by using the ALTERNATE_TIME_OFFSET_INDICATOR TLV.

In computing the TLV values currentOffset, jumpSeconds, and timeOfNextJump, the Grandmaster PTP Instance shall be responsible for including all alternate timescale offset information including, if applicable, the value of <dLS> and any pending leap seconds. If the values of the timePropertiesDS members are relevant to an alternate timescale, the Grandmaster PTP Instance shall use them to ensure correct values of currentOffset, jumpSeconds, and timeOfNextJump and must account for any offset between the timescale PTP and the timescale ARB.

The timeOfNextJump, jumpSeconds, and currentOffset shall be interpreted relative to the timescale ARB.

PTP Instances receiving the TLV shall not use the values of timePropertiesDS.currentUtcOffset, timePropertiesDS.leap59, timePropertiesDS.leap61, or the currentUtcOffset or of the leap59 and leap61 flags of a received Announce message in computing time in the alternate timescale.

The PTP Instance receiving the TLV shall add the TLV's currentOffset to the PTP Instance Time to compute time in the alternate timescale. The timeOfNextJump and jumpSeconds are used to ensure that discontinuities occur at the correct PTP Instance Time.

16.3.1.3 Discontinuities in alternate timescales

An alternate timescale may have discontinuities (jumps). For example, when the timescale is PTP, which is linear, discontinuities occur in the alternate timescale Pacific Time at the beginning and end of daylight savings time and at the time of a leap second. As another example, when the timescale is ARB, the alternate timescale with an initial 37-min offset from ARB may have discontinuities as the result of a discontinuity in the timescale ARB created by an administrative procedure (see 7.2.1).

If a discontinuity is about to occur, the Grandmaster PTP Instance shall indicate this information in a contiguous sequence of at least 256 Announce messages transmitted immediately before the PTP Instance Time of the discontinuity. This restriction is to ensure that the information is distributed early enough that it reaches all PTP Instances prior to the time of the discontinuity. The time and magnitude of this discontinuity shall be indicated using the jumpSeconds and timeOfNextJump fields specified in Table 116.

Normally, PTP Instances receive a sequence of Announce messages with the attached ALTERNATE_TIME_OFFSET_INDICATOR TLV well in advance of the time indicated by timeOfNextJump. Such PTP Instances shall ensure that in all circumstances, the offset information in the sequence of TLVs is applied only once for each distinct offset specification and, if possible, at the time indicated by the value of timeOfNextJump. However, if the jumpSeconds field of the first received ALTERNATE_TIME_OFFSET_INDICATOR TLV entity is nonzero, indicating a forthcoming discontinuity, and the PTP Instance Time of the receiving PTP Instance is greater than the value of timeOfNextJump field of the received TLV, that is, the first received notification is late, then the PTP Instance shall not implement the discontinuity correction based on any of the TLVs in the sequence. Instead, the receiving PTP Instance shall wait for the arrival of an ALTERNATE_TIME_OFFSET_INDICATOR TLV from the Grandmaster PTP Instance that contains the updated value of currentOffset and with a timeOfNextJump equal to 0 or in the future, and use the updated value of currentOffset in the determination of the time in the alternate timescale.

NOTE—In normal operation, ALTERNATE_TIME_OFFSET_INDICATOR TLVs containing the updated value of currentOffset (see 16.3.3.5) will always be received from the Grandmaster PTP Instance. The updated information can be used as a safety check on this process.

The properties of the alternate timescale offset mechanism may be managed using the alternateTimescaleOffsetsDS data set.

16.3.2 Propagation by Boundary Clocks

A Boundary Clock that is not the Grandmaster PTP Instance and that implements the alternate timescale offset option shall propagate the information contained in all ALTERNATE_TIME_OFFSET_INDICATOR TLV entities contained in the most recent Announce message received from its Master PTP Instance in any Announce message that it transmits.

NOTE—It is recommended that all Boundary Clocks in the domain between the Slave PTP Instance and its Grandmaster PTP Instance implement the alternate timescale offset option.

16.3.3 ALTERNATE_TIME_OFFSET_INDICATOR TLV specification

16.3.3.1 General

The ALTERNATE_TIME_OFFSET_INDICATOR TLV format shall be as specified in Table 116.

Table 116—ALTERNATE_TIME_OFFSET_INDICATOR TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
keyField								1	4
currentOffset								4	5
jumpSeconds								4	9
timeOfNextJump								6	13
displayName								L	19
pad								M	19+L

16.3.3.2 tlvType

The value of tlvType shall be ALTERNATE_TIME_OFFSET_INDICATOR.

16.3.3.3 lengthField

The value of the lengthField is $15 + L + M$, where L is the length of the displayName field and M is the length of the pad field.

16.3.3.4 keyField (UInteger8)

The value of keyField shall indicate the alternate timescale reported in this TLV entity.

16.3.3.5 currentOffset (Integer32)

The value of currentOffset shall be the offset of the alternate timescale, in seconds, from the PTP Instance Time defined as $\text{currentOffset} = (\text{time measured in the alternate timescale}) - \text{PTP Instance Time}$. Time in the alternate timescale is the sum of this value and the PTP Instance Time (see 3.1.54).

NOTE—With this definition, the sign convention of currentOffset is the opposite of the sign convention of currentUtcOffset (see 7.2.4).

16.3.3.6 jumpSeconds (Integer32)

The value of jumpSeconds shall be the size of the next discontinuity, in seconds, in the alternate timescale. A value of zero indicates that no discontinuity is expected. A positive value indicates that the discontinuity will cause the currentOffset of the alternate time to increase.

NOTE—The new value of currentOffset (i.e., after the discontinuity) is equal to the old value of currentOffset (i.e., before the discontinuity) plus jumpSeconds. This means that if currentOffset is negative and jumpSeconds is positive, the absolute value of currentOffset could possibly decrease.

16.3.3.7 timeOfNextJump (UInteger48)

The value of timeOfNextJump shall be the value of the seconds portion of the PTP Instance Time of the transmitting PTP Instance at the time that the next discontinuity will occur. The discontinuity occurs at the start of the second indicated by the value of timeOfNextJump.

NOTE—If the alternate timescale implements a leap-second correction (e.g., if the alternate timescale is a local timescale at the time of a leap second, or if the option of this subclause is being used to compute UTC time at the time of a leap second), timeOfNextJump is the time when $\langle \text{dLS} \rangle = \text{TAI} - \text{UTC}$ changes. This time is specified by the IERS (see 7.2.4 and IERS Bulletin C). There is no direct indication that the jump is due to the leap second when using the alternate timescales option, and so it might not be possible to derive the correct ISO 8601:2004 [B29] print form (with seconds equal to 60) for the alternate timescale at the time of the leap second.

16.3.3.8 displayName (PTPText)

The value of displayName is a textual description of the alternate timescale.

NOTE 1—Commonly used acronyms are typically used, for example, PT, PST, PDT for Pacific Time, Pacific Standard Time, and Pacific Daylight Time, respectively. The displayName can be application specific (e.g., “Launch”).

The maximum number of symbols in the displayName.textField field (see 5.3.9) shall be 10.

NOTE 2—The desired textual description can change at a discontinuity, for example, from PST to PDT. There is no mechanism to align the change of the displayName with the time of the discontinuity. Therefore the received displayName might be incorrect during transients, for example, for the short period between the timeOfNextJump and the arrival of the next ALTERNATE_TIME_OFFSET_INDICATOR TLV.

16.3.3.9 pad (Octet[M])

The pad field shall be an octet array of length M where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of 5.3.8 are met.

16.3.4 alternateTimescaleOffsetsDS data set specifications

16.3.4.1 General

The members of this data set are as follows:

- alternateTimescaleOffsetsDS.maxKey
- alternateTimescaleOffsetsDS.list

This data set is contained within the data sets of the PTP Instance (see 8.2.9).

16.3.4.2 AlternateTimescale derived data type

The AlternateTimescale data type represents each element in the data set's list, alternateTimescaleOffsetsDS.list (see 16.3.4.4.1).

```
Struct AlternateTimescale
{
  UInteger8 keyField;
  Boolean enable;
  Integer32 currentOffset;
  Integer32 jumpSeconds;
  UInteger48 timeOfNextJump;
  PTPText displayName;
};
```

16.3.4.3 Static members of the alternateTimescaleOffsetDS data set

16.3.4.3.1 alternateTimescaleOffsetDS.maxKey

The maxKey member allows management to determine the number of alternate timescales in the list (16.3.4.4.1). The data type shall be UInteger8.

The value of maxKey shall indicate the value of the largest keyField in the list.

16.3.4.4 Configurable members of the alternateTimescaleOffsetDS data set

16.3.4.4.1 alternateTimescaleOffsetDS.list

The list member provides the list of alternate timescales in the PTP Instance. The data type of each element of the list is AlternateTimescale.

Elements in the list can be created or deleted if those operations are supported by the management mechanism.

Individual items within each AlternateTimescale data type (e.g., currentOffset) can be read or written if those operations are supported by the management mechanism.

If management write is supported for AlternateTimescale items currentOffset, jumpSeconds, and

timeOfNextJump, the value for all three items shall be provided within a single write operation, and the update of all three items shall be atomic. If any of the three values fails to update, a management error shall be returned.

Subclause 16.3.4.4.1.1 through 16.3.4.4.1.6 specify the items of the AlternateTimescale data type.

16.3.4.4.1.1 AlternateTimescale.keyField

The keyField, alternateTimeOffsetDS.list[keyField] member AlternateTimescale.keyfield shall be a unique identifier of each element in the list. A management read or write of a list element (i.e., items in the AlternateTimescale data type) shall include the element's keyField as identification. If a list element corresponding to management's keyField is not found, a management error shall be returned.

16.3.4.4.1.2 AlternateTimescale.enable

If the value of alternateTimeOffsetDS.list[keyField] member AlternateTimescale.enable is TRUE, the ALTERNATE_TIME_OFFSET_INDICATOR TLV for this AlternateTimescale shall be attached to Announce messages. If enable is FALSE, the TLV shall not be attached.

16.3.4.4.1.3 AlternateTimescale.currentOffset

The value of alternateTimeOffsetDS.list[keyField] member AlternateTimescale.currentOffset shall be the offset of the alternate time from PTP Instance Time in the Grandmaster PTP Instance (see 16.3.3.5).

16.3.4.4.1.4 AlternateTimescale.jumpSeconds

The value of alternateTimeOffsetDS.list[keyField] member AlternateTimescale.jumpSeconds shall be the size of the next discontinuity (see 16.3.3.6).

16.3.4.4.1.5 AlternateTimescale.timeOfNextJump

The value of alternateTimeOffsetDS.list[keyField] member AlternateTimescale.timeOfNextJump shall be the time that the next discontinuity will occur based on PTP Instance Time in the Grandmaster PTP Instance (see 16.3.3.7).

16.3.4.4.1.6 AlternateTimescale.displayName

The value of alternateTimeOffsetDS.list[keyField] member AlternateTimescale.displayName shall be a textual description of the alternate timescale (see 16.3.3.8).

16.4 Holdover upgrade (optional)

16.4.1 General description of the holdover upgrade option

Subclause 16.4 provides a mechanism for a PTP Instance that:

- a) is not currently a Grandmaster PTP Instance, and
- b) has very good holdover capabilities, and
- c) has no direct access to the desired source of time (or of appropriate frequency) for the domain,

to potentially become the Grandmaster PTP Instance in the event, the previous Grandmaster PTP Instance is disconnected or its characteristics degrade. Such a clock is termed a “holdover-upgradable PTP Instance” for purposes of this option.

NOTE—In some applications, PTP is used to distribute frequency and not time.

For purposes of this subclause, holdover refers to the capability of a PTP Instance to deliver time (or frequency) to the domain, within application or profile specifications, after loss of the PTP Instance’s reference.

A holdover-upgradable PTP Instance shall have the following characteristics:

- The ability to synchronize its Local PTP Clock to the Grandmaster Clock selected by the BMCA
- After being synchronized can maintain, to application or profile specifications, its estimate of the time (or frequency) if required to do so by the operation of this option
- The ability to adhere to the other requirements of this subclause

16.4.1.1 holdoverUpgradeDS.enable

The data type for holdoverUpgradeDS.enable shall be Boolean. The classification is configurable.

When supported, the value TRUE shall indicate that the holdover upgrade mechanism is enabled on the PTP Port (see 16.4.3), and the value FALSE shall indicate that the holdover upgrade mechanism is disabled on the PTP Port (see 16.4.2).

16.4.2 Behavior of a holdover-upgradable PTP Instance when in the disabled state.

When the value of holdoverUpgradeDS.enable (see 16.4.1.1) is FALSE, the holdover-upgradable PTP Instance shall not use the holdover upgrade options indicated for clockClass values of 7, 14, 187, and 193 in Table 4.

16.4.3 Behavior of a holdover-upgradable PTP Instance when in the enabled state.

When the value of holdoverUpgradeDS.enable (see 16.4.1.1) is TRUE, the holdover-upgradable PTP Instance semantics are as follows.

When receiving synchronization PTP messages from a Grandmaster PTP Instance with clockClass 6 or 13 the holdover-upgradable PTP Instance shall:

- Have a clockClass value of 248, or a value in the ranges designated in Table 4 as “For use by alternate PTP Profiles” if specified by the applicable PTP Profile
- Execute the specification of the other applicable clauses of the standard based on the outcome of the applicable BMCA

When not receiving PTP synchronization messages from a Grandmaster PTP Instance with clockClass 6 or 13, the holdover-upgradable PTP Instance shall obey the specifications of the upgrade configuration policy.

The upgrade configuration policy is profile specific and shall specify the algorithm used by the holdover-upgradable PTP Instance in determining all of the following:

- a) The conditions that must exist for a change in the clockClass of the holdover-upgradable clock to occur
- b) When such a change in the clockClass is to occur
- c) The allowed upgraded values permitted for the upgraded clockClass

Upgraded values shall be either 7, 52, or 187 if the timescale PTP is in use or 14, 58, or 193 if the timescale ARB is in use, or any of the values designated in Table 4 as “For use by alternate PTP Profiles” is in use, if specified by the applicable PTP Profile.

This algorithm shall be based on the holdover capability specifications of the holdover-upgradable PTP Instance, and the advertised clockQuality of the Grandmaster PTP Instance to which it was locked.

NOTE 1—For example, an upgrade configuration policy might require the following:

- The most recent value of stepsRemoved from a Grandmaster PTP Instance with a clockClass value of 6 be less than 5
- The minimum holdover time of the holdover-upgradable PTP Instance be at least 50 ms
- It is within its holdover specification
- The clockClass not change until at least 5 ms after Sync messages from the former Grandmaster PTP Instance cease or the clockClass of the former Grandmaster Clock degrades to 187, and
- The holdover-upgradable PTP Instance’s upgraded value can only be 187

In the absence of a specification for the upgrade configuration policy, no upgrades in clockClass values shall be permitted under the terms of this option.

NOTE 2—Utilization of this holdover capability with PTP Instances using clockClass upgrade values less than 128 might cause segmentation of the network similar to when multiple clocks synchronize to a primary reference time source. Therefore, the consideration on utilization of this behavior needs to take this into account. It is expected that only a limited number of PTP Instances with superior local oscillators will have this behavior activated within a PTP domain.

16.5 Isolation of PTP Instances running under profiles specified by different standards organizations (optional)

16.5.1 General

This option is enabled if and only if the applicable PTP Profile fully implements the specifications of 16.5.2.

This option enables PTP Instances operating under a given PTP Profile specified by a QSDO to be isolated from PTP Instances operating under PTP Profiles written by a different QSDO when both are executing on the same PTP Network. The option provides isolation over and above the 2008 edition isolation, which is obtained by using the domainNumber and majorSdoId attributes alone.

Isolation in the 2008 edition is achieved by scoping based on user-configured values of the domainNumber and two values, 0 and 1, of the majorSdoId, that is, the transportSpecific field of the 2008 edition. The value 0 can be used by any PTP Profile specified by any organization, while the value 1 is restricted to the PTP Profiles developed by the IEEE 802.1 Working Group (see Table 2 of 7.1.4). Thus, for value 0 there can be PTP Instances operating under many PTP Profiles with only the user configurable domainNumber available for isolation.

Stronger isolation can be obtained based on the terms of this subclause.

16.5.2 PTP Profile specifications for this option

To create a PTP Profile providing the strong isolation described above, the QSDO writing the profile:

- Shall obtain a value of the sdoId from the IEEE RA.¹⁹ From Table 2, this value will be in the range 300₁₆ through FFC₁₆.
- Shall specify that all PTP Instances operating under the PTP Profile use this value.
- Since a QSDO can obtain only a single unique sdoId value from the IEEE RA, a QSDO writing multiple PTP Profiles under the terms of this subclause shall ensure that additional isolation measures are specified for each of the multiple PTP Profiles. Such measures may include each such PTP Profile specifying restrictions on domainNumber values tighter than those specified in Table 2, for example, if on layer 3 having one such PTP Profile use domainNumbers in the range 128 to 150 and a second such PTP Profile use domainNumbers in the range 151 to 239.

NOTE—This option provides isolation from devices that are strictly conformant to the domainNumber and sdoId specifications of this edition, and that have a different value of sdoId, and also with devices conformant to the domainNumber and transportSpecific specifications of the 2008 edition. The use of this option is not available to devices conformant to the 2008 edition unless they upgrade their implementation.

16.6 Common Mean Link Delay Service (optional)

16.6.1 General

The Common Mean Link Delay Service (CMLDS) is an optional service that enables any PTP Port that would normally obtain the value of a link's <meanLinkDelay> and <neighborRateRatio> using the peer-to-peer method to instead obtain these values from this optional service. The CMLDS service is available to all PTP Instances communicating with a specific transport mechanism, for example, using Annex E, over the physical link between two PTP Nodes.

In this option, the term “Link Port” refers to the mechanism enabling communication with a specific transport mechanism, for example, using Annex E, over the physical link between two PTP Nodes.

¹⁹ sdoIds can be obtained from the IEEE Registration Authority (<https://standards.ieee.org/regauth/>).

The Common Mean Link Delay Service is designed to run independently from any PTP Instances communicating over a Link Port. The service provides information on the `<meanLinkDelay>` as well as the `<neighborRateRatio>` measured in the timescale used by the service. The service runs on every Link Port where the CMLDS is present. Information required by a PTP Port is requested from and delivered by the service running on the associated Link Port.

How this information is transferred from the service running on a Link Port to a requesting PTP Port associated with the same Link Port is of necessity dependent on the programming and operating system environment of the hosting PTP Node. Therefore, this standard only specifies the information to be transferred but not the specific mechanism.

16.6.2 Using the Common Mean Link Delay Service.

If the CMLDS is implemented on the Link Port, and the value of `portDS.delayMechanism` for a PTP Port of a PTP Instance on the PTP Node is `COMMON_P2P`, then the PTP Port shall obtain the value of the `<meanLinkDelay>` and `<neighborRateRatio>` for a link by invoking the information transfer mechanism of the Common Mean Link Delay Service of the Link Port associated with the PTP Port whenever it normally would have issued a `Pdelay_Req` message when not using this option.

If the `serviceMeasurementValid` field of the returned information (see 16.6.3.2) is `TRUE`, then based on the returned information, the PTP Port shall convert the returned information into the timescale of the domain and use this information in computations involving the `<meanLinkDelay>` and residence times.

See 6.6.6.2 for a discussion of `<neighborRateRatio>` and `<cumulativeRateRatio>`.

If the `serviceMeasurementValid` field of the returned information is `FALSE`, or if no return is received within a manufacturer specified time interval, then the PTP Port may revert to using the normal peer to peer delay mechanism specified in 11.4. Applicable PTP Profiles should specify any needed timeouts or mechanisms to handle failure to receive the information or for retrying the use of the Common Mean Link Delay Service.

16.6.3 Common Mean Link Delay Service

16.6.3.1 General CMLDS specifications

If the CMLDS service is implemented on the Link Port, then the CMLDS service shall execute as specified in 16.6.3.

The service shall use values of `sdoId` and `domainNumber` as follows:

- a) If executing on layer2 IEEE 802 (i.e., Annex E), `sdoId` is `20016` and the `domainNumber` is 0.
- b) If not executing on layer2 IEEE 802 (i.e., Annex E), `sdoId` is `20016` and the `domainNumber` is 128.

NOTE 1—`sdoId` value `20016` is owned by the IEEE 1588 Working Group. `sdoId` `20016` corresponds to values of `majorSdoId` = `216` and `minorSdoId` = `0016`.

NOTE 2—A failure to run this service on all links can result in a PTP Instance not having an estimate of `<meanLinkDelay>` nor the `<neighborRateRatio>` on such Link Ports.

For each Link Port, the service shall respond to a received `Pdelay_Req` message with a `Pdelay_Resp` message, and a `Pdelay_Resp_Follow_Up` message, as specified for the peer-to-peer delay mechanism in 11.4. In all cases the Link Port responding to a received `Pdelay_Req` message shall operate as a two-step PTP Port for peer-to-peer delay mechanism messages and therefore must use the specification of 11.4.2

choice “c)” for two-step ports and within choice “c)” use choice “0” where the timestamps t_2 and t_3 are transmitted in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.

NOTE 3—The reception of a Pdelay_Req message only occurs if both the sender and the receiver are using the same values of sdoId and domainNumber, for example, sdoId = 200₁₆ and domainNumber 0.

16.6.3.2 CMLDS link measurement operation

For each Link Port, if both the value of portDS.portEnable is TRUE and the value of portDS.delayMechanism is COMMON_P2P on any PTP Port communicating via the Link Port, then the following specifications of this subclause shall be in effect, otherwise the CMLDS service shall not execute the following specifications of this subclause.

The service shall measure the <meanLinkDelay> and the <neighborRateRatio> using the peer-to-peer delay mechanism and multicast communications as specified in 11.4 and the specifications of 16.10 for the computation of <neighborRateRatio>. The measurements shall be based on a Local Clock accessible to all PTP Instances communicating via the Link Port. The measurement shall be made at a rate defined as follows:

- The initial Pdelay_Req message may be transmitted when required.
- Subsequent Pdelay_Req messages shall be transmitted at the rate indicated by the value of the data set member cmlDsLinkPortDS.logMinPdelayRequestInterval.

The values for the fields of the Common Header of the Pdelay_Req messages transmitted by the service shall be as defined in 13.3.1 and 13.3.2 with the following exceptions:

- a) **majorSdoId:** The value shall be as defined in 16.6.3.1.
- b) **minorVersionPTP:** The value shall be the value of cmlDsLinkPortDS.minorVersionNumber (see 16.6.4.2.4.3).
- c) **versionPTP:** The value shall be the value of cmlDsLinkPortDS.versionNumber (see 16.6.4.2.4.2).
- d) **domainNumber:** The value shall be as defined in 16.6.3.1.
- e) **minorSdoId:** The value shall be as defined in 16.6.3.1.
- f) **flagField:** The values shall be as specified in Table 37 except for the values of PTP Profile Specific 1 and PTP Profile Specific 2, both of which shall be FALSE.
- g) **sourcePortIdentity:** The value shall be the value of cmlDsLinkPortDS.portIdentity (see 16.6.4.2.2.1).

The measurement of the <meanLinkDelay> and the <neighborRateRatio> must be sufficiently accurate to enable users of the service to correct their computations to the accuracy required by the PTP Profiles applicable to the users.

Upon receipt of a request for information from the Common Mean Link Delay Service, the service shall provide the information specified below to the requesting PTP Port.

The returned information shall be represented with the data type defined as follows:

```
Struct CommonMeanLinkDelayInformation
{
  Boolean serviceMeasurementValid;
  TimeInterval meanLinkDelay;
  Integer32 scaledNeighborRateRatio;
}
```

where

- h) The value of serviceMeasurementValid shall be as follows:
 - 1) FALSE if (a) no Pdelay_Resp or Pdelay_Resp_Follow_Up message is received in response to a Pdelay_Req message issued by the service, or (b) multiple Pdelay_Resp or multiple Pdelay_Resp_Follow_Up messages are received in response to a single Pdelay_Req message issued by the service or (c) any other failure conditions specified by the applicable PTP Profile are detected, otherwise,
 - 2) TRUE.
- i) The value of <meanLinkDelay> shall be the measured value for the Link Port.
- j) The value of <neighborRateRatio> shall be computed based on the peer-to-peer delay mechanism measurements as specified in 11.4.2 and the specifications of 16.10 for the computation of <neighborRateRatio>. The scaledNeighborRateRatio is computed as follows:

$\text{scaledNeighborRateRatio} = (\text{<neighborRateRatio>} - 1) \times 2^{41}$ and truncated to the next smaller signed integer.

NOTE 1—This scaling allows representation of fractional frequency offsets in the approximate range $[-9.766 \times 10^{-4}, 9.766 \times 10^{-4}]$ and with a granularity of 2^{-41} .

Upon receipt of a request for information, the Common Mean Link Delay Service may in addition return the raw measurement data gathered by the service for use in estimating the <meanLinkDelay> and <neighborRateRatio>.

NOTE 2—By returning the raw measurement data, users of the service are able to apply their own algorithms for estimating <meanLinkDelay> and <neighborRateRatio> to meet application of PTP Profile requirements.

16.6.4 commonMeanLinkDelayServiceDS data set specifications

If the optional Common Mean Link Delay Service is supported, the required common MeanLinkDelayService data sets shall be supported and the optional commonMeanLinkDelayService data sets may be supported. The operational conformance of these data sets is shown in Table 117.

Table 117—commonMeanLinkDelayService operational conformance

Name	Required/Optional
cmldsDefaultDS	required
cmldsLinkPortDS	required
cmldsTimestampCorrectionLinkPortDS	optional
cmldsAsymmetryCorrectionLinkPortDS	optional
cmldsPerformanceMonitoringLinkPortDS	optional

In the hierarchy of data sets (see 8.1.4.2), the data sets for common services follow the data sets for PTP Instances and Transparent Clocks. Each common service is under commonServices. For the optional commonMeanLinkDelayService, two required data sets and three optional data sets are defined. Link Ports (see 16.6.1) of the Common Mean Link Delay Service may be created or deleted dynamically in implementations that support dynamic create/delete of devices.

The commonMeanLinkDelayService contains the cmldsLinkPortList, which is a list of CMLDS Link Ports of the PTP Node. There needs to be a Link Port (i.e., a CMLDS must be present) that is available to every PTP port that can use the CMLDS (i.e., where portDS.delayMechanism of that PTP instance can have the value COMMON_P2P).

The Common Mean Link Delay Service Data Sets are not maintained separately for each PTP Instance.

Rather, a single copy of the `cmldsDefaultDS` is maintained for the PTP Node, and a single copy of each data set under the `cmldsLinkPortList` is maintained per Link Port of the CMLDS of the PTP Node.

There is one optional `commonServicesPortDS` for each PTP Port of each PTP Instance. This data set enables a PTP Port of a PTP Instance to determine which port of the respective common service corresponds to that PTP Port. For the CMLDS, and the only member of the `commonServicesPortDS` is the `cmldsLinkPortPortNumber`. This member contains the port number of the CMLDS Link Port that corresponds to this PTP Port.

16.6.4.1 `cmldsDefaultDS`

16.6.4.1.1 General

The name and operational conformance (see 8.1.6) of each `cmldsDefaultDS` member is summarized in Table 118.

Table 118—`cmldsDefaultDS` operational conformance

Name	Required/Optional
<code>clockIdentity</code>	required
<code>numberLinkPorts</code>	required

16.6.4.1.2 Static members of the `cmldsDefaultDS`

16.6.4.1.2.1 `cmldsDefaultDS.clockIdentity`

The value of `cmldsDefaultDS.clockIdentity` shall be the `clockIdentity` (see 7.6.2.2) of the Common Mean Link Delay Service. The data type shall be `ClockIdentity`.

NOTE—`cmldsDefaultDS.clockIdentity` is distinct from the `defaultDS.clockIdentity` of any PTP Instance of the same node.

16.6.4.1.2.2 `cmldsDefaultDS.numberLinkPorts`

The value of `cmldsDefaultDS.numberLinkPorts` shall be the number of Link Ports of a PTP Instance executing the Common Mean Link Delay Service (see 16.6.1). The data type shall be `UInteger16`.

NOTE—The value of `cmldsDefaultDS.numberLinkPorts` is not necessarily the same as the value of `defaultDS.numberLinkPorts` for any PTP Instance of the same PTP Node.

16.6.4.2 `cmldsLinkPortDS`

16.6.4.2.1 General

Each `cmldsLinkPortDS` corresponds to a single Link Port of the Common Mean Link Delay Service. The name and operational conformance (see 8.1.6) of each `cmldsLinkPortDS` member is summarized in Table 119.

Table 119—cmlDsLinkPortDS operational conformance

Name	Required/Optional
portIdentity	Required
commonMeanLinkDelayInformation	Required
logMinPdelayReqInterval	Required
versionNumber	Required
minorVersionNumber	Required
delayAsymmetry	Required
domainNumber	Required

16.6.4.2.2 Static members of the cmlDsLinkPortDS

16.6.4.2.2.1 cmlDsLinkPortDS.portIdentity

The value of cmlDsLinkPortDS.portIdentity shall be the PortIdentity attribute of the Link Port (see 7.5.2). The data type shall be PortIdentity. The value of cmlDsLinkPortDS.portIdentity.clockIdentity shall be the clockIdentity of the Common Mean Link Delay Service (see 16.6.4.1.2.1). The value of cmlDsLinkPortDS.portIdentity.portNumber shall be unique to the Link Port.

16.6.4.2.2.2 cmlDsLinkPortDS.domainNumber

The value of cmlDsLinkPortDS.domainNumber shall be the domainNumber value used by CMLDS, as specified in 16.6.3.1.

16.6.4.2.3 Dynamic members of the cmlDsLinkPortDS

16.6.4.2.3.1 cmlDsLinkPortDS.commonMeanLinkDelayInformation

The value of cmlDsLinkPortDS.commonMeanLinkDelayInformation shall be the CommonMeanLinkDelay Information structure returned by the Common Mean Link Delay Service (see 16.6.2 and 16.6.3). The data type shall be CommonMeanLinkDelayInformation. The initialization value of cmlDsLinkPortDS.commonMeanLinkDelayInformation.serviceMeasurementValid shall be FALSE. The initialization value of cmlDsLinkPortDS.commonMeanLinkDelayInformation.meanLinkDelay shall be 0. The initialization value of cmlDsLinkPortDS.commonMeanLinkDelayInformation.scaledNeighborRateRatio shall be 0.

16.6.4.2.4 Configurable members of the cmlDsLinkPortDS

16.6.4.2.4.1 cmlDsLinkPortDS.logMinPdelayReqInterval

The value of cmlDsLinkPortDS.logMinPdelayReqInterval shall be the logarithm to the base 2 of the minPdelayReqInterval (see 7.7.2.5). The data type shall be Integer8.

16.6.4.2.4.2 cmlDsLinkPortDS.versionNumber

The value of cmlDsLinkPortDS.versionNumber shall indicate the PTP version in use on the Common Mean Link Delay Service Link Port. The data type shall be UInteger4.

16.6.4.2.4.3 cmldsLinkPortDS.minorVersionNumber

The value of cmldsLinkPortDS.minorVersionNumber shall indicate the PTP minor version in use on the Common Mean Link Delay Service Link Port. The data shall be UInteger4.

16.6.4.2.4.4 cmldsLinkPortDS.delayAsymmetry

The value of cmldsLinkPortDS.delayAsymmetry shall be the value of <delayAsymmetry> applicable to the Link Port as specified in 7.4.2. The data type shall be TimeInterval. The specification initialization value (see 8.1.3.4) shall be zero.

If the option of 16.8 is implemented, cmldsLinkPortDS.delayAsymmetry shall be dynamic. If the option of 16.8 is not implemented, cmldsLinkPortDS.delayAsymmetry may be configurable, that is, it may be written by any management mechanism. Otherwise, if the option of 16.8 is not implemented and portDS.delayAsymmetry is not configurable, it shall be static. If cmldsLinkPortDS.delayAsymmetry is configurable, the value of <delayAsymmetry> shall be the value of cmldsLinkPortDS.delayAsymmetry, instead of portDS.delayAsymmetry as specified in item a) of 7.4.2.

16.6.4.3 cmldsTimestampCorrectionLinkPortDS

This optional data set provides access to the configurable correction of generated timestamps (see 16.7). The cmldsTimestampCorrectionLinkPortDS applies to the Common Mean Link Delay Service Link Port.

16.6.4.3.1 cmldsTimestampCorrectionLinkPortDS.egressLatency

The specifications for this member are the same as those for timestampCorrectionPortDS.egressLatency (see 8.2.16.2).

16.6.4.3.2 cmldsTimestampCorrectionLinkPortDS.ingressLatency

The specifications for this member are the same as those for timestampCorrectionPortDS.ingressLatency (see 8.2.16.3).

16.6.4.4 cmldsAsymmetryCorrectionLinkPortDS

This optional data set provides access to delay asymmetry correction (see 16.8). The cmldsAsymmetryCorrectionLinkPortDS applies to the Common Mean Link Delay Service Link Port.

16.6.4.4.1 cmldsAsymmetryCorrectionLinkPortDS.constantAsymmetry

The specifications for this member are the same as those for asymmetryCorrectionPortDS.constantAsymmetry (see 8.2.17.2).

16.6.4.4.2 cmldsAsymmetryCorrectionLinkPortDS.delayCoefficient

The specifications for this member are the same as those for asymmetryCorrectionPortDS.scaledDelayCoefficient (see 8.2.17.3).

16.6.4.4.3 cmldsAsymmetryCorrectionLinkPortDS.enable

The specifications for this member are the same as those for asymmetryCorrectionPortDS.enable (see 8.2.17.4). The default value shall be FALSE.

16.6.4.5 cmldsPerformanceMonitoringLinkPortDS

This data set provides management access to the cmldsPerformanceMonitoringLinkPortDS of the optional performance monitoring feature (see Annex J). The recordListPeerDelay and the recordList are the members of the cmldsPerformanceMonitoringLinkPortDS. The specifications for the members of the cmldsPerformanceMonitoringLinkPortDS are the same as those for performanceMonitoringPortDS, as specified in J.5.2.1 and J.5.2.2.

16.6.5 commonServicesPortDS

16.6.5.1 General

This data set allows a PTP port of a PTP Instance to determine which common service ports correspond to the PTP Port. In this edition of the standard, this data set has one member, which corresponds to the CMLDS. Additional members will be added to this data set if needed and if and when additional common services are specified. The name and operational conformance, as defined in 8.1.6, of each commonServicesPortDS member is summarized in Table 120.

Table 120—commonServicesPortDS operational conformance

Name	Required/Optional
cmldsLinkPortPortNumber	Required

16.6.5.1.1 Static members of the commonServicesPortDS

16.6.5.1.1.1 commonServicesPortDS.cmldsLinkPortPortNumber

The value of commonServicesPortDS.cmldsLinkPortPortNumber shall be the portNumber attribute of the cmldsLinkPortDS.portIdentity (see 16.6.4.2.2.1) of the Link Port that corresponds to this PTP Port.

16.7 Configurable correction of timestamps (optional)

16.7.1 General

When this option is implemented (see 6.1), then the option shall operate as specified in this subclause.

This subclause mandates correction of timestamps per 7.3.4.2 using the values of the appropriate optional data sets members.

When this option is implemented all of the following apply:

- The data set members timestampCorrectionPortDS.egressLatency (see 8.2.16.2) and timestampCorrectionPortDS.ingressLatency (see 8.2.16.3) shall be supported, and

- The value of the `<egressLatency>` (see 7.3.4.2) shall be the value of `timestampCorrectionPortDS.egressLatency`, and
- The value of the `<ingressLatency>` (see 7.3.4.2) shall be the value of `timestampCorrectionPortDS.ingressLatency`, and
- The corrections specified in 7.3.4.2 shall be made using the values provided in these optional data sets members, making any necessary conversions of units.

NOTE 1—The values of `timestampCorrectionPortDS.egressLatency` and `timestampCorrectionPortDS.ingressLatency` are defined as `TimeInterval`, expressed in units of 2^{-16} ns. Therefore, these values need to be divided by 2^{+16} to obtain latency in nanoseconds.

NOTE 2—It is possible that the implementation-specific mechanism that captures the `<egressProvidedTimestamp>` and `<ingressProvidedTimestamp>` values has already partially corrected them for latencies (see 7.3.4.2).

NOTE 3—The measurement of `<egressLatency>` and `<ingressLatency>` are implementation specific. Annex N provides procedures for relative calibration of `<egressLatency>` and `<ingressLatency>`.

16.8 Calculation of the `<delayAsymmetry>` for certain media (optional)

16.8.1 General

When this option is both implemented (see 6.1), and enabled (see 16.8.2), then the option shall operate as specified in this subclause.

This subclause specifies the calculation of the `<delayAsymmetry>` (see 7.4.2) on a Direct PTP Link (see 3.1.8) between two PTP Instances connected using an applicable bidirectional medium that can be described using `<delayCoefficient>` (α) (see 7.4.3).

16.8.2 Enabling the option

The calculation of the `<delayAsymmetry>` for certain media of 16.8.3 can be enabled or disabled by management of the data set member `asymmetryCorrectionPortDS.enable` (see 8.2.17.4). By default, this mechanism shall be disabled unless otherwise specified in the applicable PTP Profile.

NOTE—The calculated value of `<delayAsymmetry>` is stored in `portDS.delayAsymmetry`; see 7.4.2.

16.8.3 Calculation of the `<delayAsymmetry>`

When this option is enabled, all the following apply:

- The data set `asymmetryCorrectionPortDS` (see 8.2.17) must be implemented (see 6.1)
- The value of the `<delayCoefficient>` (see 7.4.3) shall be provided by the `asymmetryCorrectionPortDS.scaledDelayCoefficient` (see 8.2.17.3)
- The value of the `<delayAsymmetry>` shall be calculated as specified in this subclause and stored in `portDS.delayAsymmetry`

NOTE 1—The `asymmetryCorrectionPortDS.scaledDelayCoefficient` data type is `RelativeDifference` (see 5.3.11); it needs to be divided by 2^{+62} to obtain the value of `<delayCoefficient>` (α) that is required in the calculations.

NOTE 2—A change of the value of `<delayAsymmetry>` during the execution of the delay mechanisms specified in 11.3 and 11.4 might adversely affect performance.

The applicable bidirectional medium²⁰ shall have all the following characteristics:

- The transmission time of the PTP event messages over the medium from the Master PTP Instance to the Slave PTP Instance (t_{ms} or $t_{resp-to-req}$) has a nearly constant relation to the transmission time of the PTP event messages over the same medium from the Slave PTP Instance to the Master PTP Instance (t_{sm} or $t_{req-to-resp}$)
- The transmission of the PTP event messages over the medium between the two PTP Instances takes place over the same physical link, for example, copper or fiber link

NOTE 3—These characteristics effectively define the term “applicable medium.”

For the applicable media, the <delayCoefficient> (α) is the relative difference between the two transmission times, t_{ms} and t_{sm} , or $t_{resp-to-req}$ and $t_{req-to-resp}$, which is defined in subclause 7.4.3 by $t_{ms} = (1 + \alpha) \times t_{sm}$ and $t_{resp-to-req} = (1 + \alpha) \times t_{req-to-resp}$ for the delay request-response and the peer-to-peer delay mechanisms, respectively.

On a Direct PTP Link (see 3.1.8) between two PTP Ports that correct their timestamps per 7.3.4, the calculated <meanDelay> = $[(t_2 - t_1) + (t_4 - t_3)] / 2 = [(t_2 - t_3) + (t_4 - t_1)] / 2$ is the delay introduced exclusively by the medium.

It is assumed that, if needed, the ingress and egress latencies are corrected with sufficient accuracy on both PTP Ports to make their contribution to the value of <delayAsymmetry> negligible (see 7.3.4.2 and 7.4.3).

On such a Direct PTP Link that uses an applicable medium, the <delayAsymmetry> shall be calculated as follows:

$$\begin{aligned} <delayAsymmetry> = asymmetryCorrectionPortDS.constantAsymmetry + [\alpha/(\alpha+2)] \\ &\times (<meanDelay>) \end{aligned}$$

where

- $\alpha = asymmetryCorrectionPortDs.scaledDelayCoefficient/2^{+62}$ (see 8.2.17.3).
- If the PTP Port is configured to use the delay request-response mechanism, then the <meanDelay> shall be the <meanPathDelay> specified in 11.3.
- If the PTP Port is configured to use the peer-to-peer delay mechanism, then the <meanDelay> shall be the <meanLinkDelay> specified in 11.4.

To obtain high accuracy of synchronization, the computation of <delayAsymmetry> should be performed each time the value of <meanDelay> is updated. When this computation is performed, the computed value of <delayAsymmetry> shall be used in the operation of the protocol and stored in portDS.delayAsymmetry.

NOTE 4—The changes in <delayAsymmetry> resulting from changes in the value of <meanDelay> are possibly negligible for applications where high accuracy is not required.

NOTE 5—Because the dynamic calculation of <delayAsymmetry> influences the adjustment of the Local PTP Clock, these calculations might need to be taken into account in the servo design.

NOTE 6—If the calculated <delayAsymmetry> does not fully compensate the asymmetry introduced by the medium when the asymmetryCorrectionPortDS.constantAsymmetry is set to zero, the asymmetryCorrectionPortDS.constantAsymmetry in the equation allows further adjustment. In such case, the value of asymmetryCorrectionPortDS.constantAsymmetry is obtained through measurement that is outside the scope of this standard.

The calculated <delayAsymmetry> corresponds to medium asymmetry (see 7.4.3). The direction of this asymmetry is reflected by the sign (positive/negative) of the value of the <delayAsymmetry> and depends

²⁰ One known example is single-mode fiber used for two-way communication; see 1000BASE-BX10 defined in IEEE Std 802.3.

on the direction of the medium (and its asymmetry) with respect to the PTP Port of the PTP Instance on which the `<delayAsymmetry>` is calculated.

NOTE 7—In other words, when a medium is connected with its connector A to a PTP Port in the Master state (or responder) of PTP Instance A, and with its connector B to a PTP Port in the Slave state (or requestor) of PTP Instance B, the `<delayAsymmetry>` will have value X. When the medium direction is inverted and the same medium is connected with its connector B to the PTP Port in the Master state (or responder) of PTP Instance A, and with its connector A to the PTP Port in the Slave state (or requestor) of PTP Instance B, the `<delayAsymmetry>` will have the value $-X$ (minus X).

Similarly to `<delayAsymmetry>`, the value of the `<delayCoefficient>` (α) depends on the direction of the medium with respect to the PTP Port of the PTP Instance that calculates the `<delayAsymmetry>`. Therefore, the value of `asymmetryCorrectionPortDs.scaledDelayCoefficient` needs to be configured accordingly for each PTP Port on the Direct PTP Link (see 7.4.3).

NOTE 8—The value of the `<delayCoefficient>` (α) can be calibrated for each direction of a particular medium (see N.4.5). However, if `<delayCoefficient>` (α) allowing calculation of `<delayAsymmetry>` for one direction of medium asymmetry is known, the `<delayCoefficient>` (α') to calculate the `<delayAsymmetry>` for the opposite direction of medium asymmetry can be calculated as follows: $\alpha' = -\alpha / (1 + \alpha)$ (see NOTE 3— in 7.4.3).

NOTE 9—For some media, it is possible to provide autodetection of the type and/or direction of the medium that is connected to a PTP Port (see NOTE 4— in 7.4.3).

16.9 Mixed multicast/unicast operation (optional)

16.9.1 General

When this option is implemented (see 6.1.1), then the option shall operate as specified in this subclause.

Mixed multicast/unicast operation may be used by Ordinary Clocks, Boundary Clocks and Transparent Clocks when `Delay_Req` and `Delay_Resp` messages are used in the measurement of the network propagation delay.

NOTE 1—The purpose is to eliminate handling of `Delay_Req`, and `Delay_Resp` messages at PTP Ports which are listening to the PTP multicast addresses but are not the intended recipient of the PTP messages.

In a mixed multicast/unicast operation, `Announce` messages shall be sent as multicast messages. `Delay_Req` and `Delay_Resp` messages shall be sent as unicast PTP messages. `Sync` and, if applicable, `Follow_Up` messages may be sent as either multicast or unicast PTP messages. Each `Follow_Up` message shall be sent in the same communication type as the corresponding `Sync` message.

If this option is used, it should be implemented on all Ordinary Clocks and Boundary Clocks.

NOTE 2—The inclusion of Transparent Clocks using this option is out of scope. A Transparent Clock does not initiate `Announce` messages and can therefore not advertise its capabilities. A Transparent Clock will process the PTP messages as specified in Clause 10.

NOTE 3—The `unicastFlag` of the PTP message common header indicates whether the message is sent as multicast or unicast (see 13.3.2.8).

Ports in the MASTER state may require unicast negotiation before responding to unicast `Delay_Req` messages. Ports in the MASTER state may append a `PORT_COMMUNICATION_AVAILABILITY_TLV` to the `Announce` message to advertise which types of communication modes it can work in (see 16.9.2.1).

If a PTP Port in the MASTER state that is not unicast capable with respect to `Delay_Resp` messages receives a unicast `Delay_Req` message, it shall ignore the `Delay_Req` message.

Ports in the MASTER state may attach a `PROTOCOL_ADDRESS` TLV to the Announce message, which contains the protocol address of the Master PTP Instance (see 16.9.2.2).

16.9.2 Port property TLVs (Optional)

16.9.2.1 `PORT_COMMUNICATION_AVAILABILITY` TLV (Optional)

The `PORT_COMMUNICATION_AVAILABILITY` TLV (see Table 121) lists the communication capabilities of a PTP Port when in the MASTER state or when sending PTP messages with the `alternateMasterFlag` set in the common header. The `PORT_COMMUNICATION_AVAILABILITY` TLV is a nonpropagating TLV. Unless the communication capabilities are specified-by-design in the applicable PTP profile, the TLV shall be appended to a multicast Announce message to allow the PTP Port to advertise its communication capabilities.

NOTE—If the PTP Network contains Transparent Clocks, the `PROTOCOL_ADDRESS` TLV is needed since, for PTP messages between the Master PTP Instance and Slave PTP Instance via an intervening Transparent Clock, the source protocol address of the transport header is changed to the source protocol address of the Transparent Clock on retransmission.

Table 121—`PORT_COMMUNICATION_AVAILABILITY` TLV format

Bits								Octets	TLV Offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
Reserved				syncMessageAvailability				1	4
Reserved				delayRespMessageAvailability				1	5

When using the option of 16.9, this TLV shall only be sent when a PTP Port is in the MASTER state, or when the `alternateMasterFlag` field of the common header is set. This TLV shall only be sent when the delay request-response mechanism is used by the PTP Port. The PTP Port shall be able to process `Delay_Req` messages received using the advertised communication modes of the PTP Port.

16.9.2.1.1 tlvType

The value of `tlvType` is `PORT_COMMUNICATION_AVAILABILITY` (see Table 52).

16.9.2.1.2 lengthField

The value of `lengthField` is 2 (see 5.3.8).

16.9.2.1.3 syncMessageAvailability (Boolean[4])

For the Sync capabilities fields, the following holds for sending Sync and, if needed, Follow_Up messages:

- Bit 0 indicates multicast capable and shall be the value of `communicationCapabilitiesPortDS.syncCapabilities.multicastCapable`.
- Bit 1 indicates unicast capable and shall be the value of `communicationCapabilitiesPortDS.syncCapabilities.unicastCapable`.
- Bit 2 indicates unicast negotiation capable and shall be TRUE if and only if both the values of

communicationCapabilitiesPortDS.syncCapabilities.unicastNegotiationCapable and unicastNegotiationPortDS.enable (see 8.2.19.2) are TRUE, otherwise it shall be FALSE.

- Bit 3 indicates unicast negotiation required and shall be the value of communicationCapabilitiesPortDS.syncCapabilities.unicastNegotiationCapable.

16.9.2.1.4 delayRespMessageAvailability (Boolean[4])

For the Delay_Resp capabilities fields, the following holds:

- Bit 0 indicates multicast capable and shall be the value of communicationCapabilitiesPortDS.delayRespCapabilities.multicastCapable.
- Bit 1 indicates unicast capable and shall be the value of communicationCapabilitiesPortDS.delayRespCapabilities.unicastCapable.
- Bit 2 indicates unicast negotiation capable and shall be TRUE if and only if both the values of communicationCapabilitiesPortDS.delayRespCapabilities.unicastNegotiationCapable and unicastNegotiationPortDS.enable (see 8.2.19.2) are TRUE; otherwise it shall be FALSE.
- Bit 3 indicates unicast negotiation required and shall be the value of communicationCapabilitiesPortDS.delayRespCapabilities.unicastNegotiationCapable.

16.9.2.2 PROTOCOL_ADDRESS TLV

The PROTOCOL_ADDRESS TLV (see Table 122) is a TLV that refers to a specific PTP Port. It includes the protocol type and address of the PTP Port. The PROTOCOL_ADDRESS TLV is a nonpropagating TLV. It is appended to a multicast Announce message to allow a PTP Port in the MASTER state or with the alternateMaster flag set to advertise its protocol address.

NOTE—The TLV is needed since, for PTP messages between the Master PTP Instance and Slave PTP Instance via an intervening Transparent Clock, the source protocol address of the transport header is changed to the source protocol address of the Transparent Clock on retransmission.

Table 122—PROTOCOL_ADDRESS TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
portProtocolAddress								4 + N	4
Pad								M	8+N

16.9.2.2.1 tlvType

The tlvType is PROTOCOL_ADDRESS (see Table 52).

16.9.2.2.2 lengthField

The value of lengthField is 4 + N + M, where N is the size in octets of the addressField member of the portAddress type, that is, addressLength (see 5.3.6) and M is either 0 or 1 to make 4 + N + M an even number (see 5.3.8).

16.9.2.2.3 portProtocolAddress (PortAddress)

The value of the portProtocolAddress field is the protocol address of the PTP Port, which sends the PTP message with this TLV attached. The value shall be the protocolAddress, which is used as the source address by the network transport protocol for this PTP Port.

NOTE—The value of the protocol address of the sending PTP Port is obtained from the optional data set member descriptionPortDS.protocolAddress (see 8.2.18.3) if this member is implemented.

16.9.2.2.4 pad (Octet[M])

The pad field shall be an octet array of length M where M is either 1 or 0. If M is 1, all bits in the octet shall be 0. The value of M shall be such that the requirements of are 5.3.8 met.

16.10 Cumulative frequency transfer method for synchronizing clocks (optional)

16.10.1 General

This subclause specifies a mechanism for synchronizing PTP Instances as an alternative to the peer-to-peer delay mechanism specifications of Clause 10 and Clause 11. The use of this option must be specified in the applicable PTP Profile. All PTP Instances in a domain must use the option, and shall operate as specified in this subclause. The mechanism is a variant on the use of the peer-to-peer delay mechanism.

The mechanism can be implemented in Boundary Clocks, Ordinary Clocks, and peer-to-peer Transparent Clocks.

The mechanism must not be used in portions of a PTP Network that use the delay request-response mechanism.

16.10.2 Operation of the peer-to-peer delay mechanism using this option

Each PTP Port in the domain shall:

If specified in the applicable PTP Profile, obtain values for the <meanLinkDelay> and <neighborRateRatio> from the Common Mean Link Delay Service of 16.6, else

- a) For Boundary Clocks or Ordinary Clocks, execute the two-step peer-to-peer delay mechanism specifications of 11.4.2 with the modifications specified in this subclause, or
- b) For Transparent Clocks, execute the two-step peer-to-peer delay mechanism specifications of 10.3 with the modifications specified in this subclause.

In all cases the responder shall operate as a two-step PTP Port for peer-to-peer delay mechanism messages and therefore must use the specification of choice “c)” in 11.4.2.

NOTE 1—This choice enables the requester to obtain the values of t_2 and t_3 needed to compute the neighborRateRatio.

NOTE 2—This specification requires that PTP Instances and the Common Mean Link Delay Service use two-step semantics for peer-to-peer delay messages (see 7.5.2.5).

This procedure results in each PTP Port measuring a domain-specific value for the <meanLinkDelay> and

the <neighborRateRatio> between it and the PTP Port at the other end of the PTP-Link.

If the Common Mean Link Delay Service is used, the PTP Port can also obtain values for both the <meanLinkDelay> and the <neighborRateRatio> for the PTP Link.

NOTE 3—See 11.4.3 for restrictions on the use of the peer-to-peer delay mechanism.

In each PTP Instance and in the Common Mean Link Delay Service serving the PTP Ports of the PTP Instance, the Timestamping Clock for Sync, Pdelay_Req, and Pdelay_Resp messages on both ingress and egress shall be one of the following:

- c) The Local Clock for all Transparent Clocks and for all Boundary Clocks that are not the Grandmaster Clock of the domain, as well as the Local PTP Clock for the Grandmaster Clock and any Ordinary Clocks
- d) Some other selection that uses other choices as permitted by 7.3.4.3 provided appropriate corrections are applied

NOTE 4—For item d), the applicable PTP Profile needs to specify these corrections as well as additional specifications for 16.10.4.

The remaining specifications of this option are predicated on using choice c).

NOTE 5—Future editions can provide additional specifications for option “d)” and the above statement permits this to be done and still be conformant to this subclause.

In addition to computing the <meanLinkDelay> (see 16.10.4.2), the PTP Port shall compute the <neighborRateRatio>, which is the ratio of the frequency of the Timestamping Clock of the responding PTP Port to the frequency of the Timestamping Clock of the requesting PTP Port.

NOTE 6—In general the values of the <meanLinkDelay> and <neighborRateRatio> will be different on each PTP Port and in each domain and in the Common Mean Link Delay Service.

The <neighborRateRatio> shall be computed using the values of the following information associated with successive peer-to-peer delay message exchanges:

- e) The <pdelayReqEventEgressTimestamp> at the requester (i.e., t_1 , see Figure 42).
- f) The <pdelayReqEventIngressTimestamp> at the requester (i.e., t_4 , see Figure 42).
- g) The requestReceiptTimestamp field and correctionField of the Pdelay_Resp message received at the requester. These fields can be combined, by subtracting with appropriate unit conversions, to obtain t_2 (see 11.4.2c)0 and Figure 42).
- h) The responseOriginTimestamp field and correctionField of the Pdelay_Resp_Follow_Up message received at the requester. These fields can be combined, by subtracting with appropriate unit conversions, to obtain t_3 (see item c)0 in 11.4.2 and Figure 42).

A PTP Profile may specify requirements, for example, accuracy, and methods (e.g., see NOTE 7), for computing the value of <neighborRateRatio> based on the above timestamps.

NOTE 7—As one example, the <neighborRateRatio> can be estimated as the ratio of the elapsed time of the Timestamping Clock of the PTP Instance at the other end of the link attached to this PTP Port, to the elapsed time of the Timestamping Clock of this PTP Instance. This ratio can be computed for the time interval between a set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and a second set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages N peer-to-peer delay message exchanges later, that is, as shown in Equation (7).

$$< \text{neighborRateRatio} > = \frac{(t_3)_N - (t_3)_0}{(t_4)_N - (t_4)_0} \quad (7)$$

where the peer-to-peer delay exchanges are indexed from 0 to N.

16.10.3 Generating the <cumulativeRateRatio>

The Grandmaster Clock of the domain shall attach the CUMULATIVE_RATE_RATIO TLV (see Table 123) to (a) the Sync message if transmitting from a one-step PTP Port or (b) either the Sync message or the Follow_Up message if transmitting from a two-step PTP Port. The value of the cumulativeRateRatio field of the TLV attached by the Grandmaster Clock shall be as follows:

- a) If an external Clock Source is selected by the source selection algorithm of the External Source block (see 7.6.6.3), then the value shall be as follows:
 - 1) One if the Timestamping Clock of the Grandmaster Clock is syntonized to the selected external Clock Source
 - 2) The ratio of the frequency of the Timestamping Clock to the frequency of the external Clock Source if the Instance Timestamping Clock of the Grandmaster Clock is not syntonized to the selected external Clock Source
- b) One if the Timestamping Clock of the Grandmaster Clock was selected by the source selection algorithm of the External Source block (see 7.6.6.3)

All Boundary and Transparent Clocks in the domain receiving the CUMULATIVE_RATE_RATIO TLV shall multiply the value of the cumulativeRateRatio field of the received TLV by the value of the <neighborRateRatio> pertaining to the ingress PTP Port measured per 16.10.2. This computed value shall be the value of the cumulativeRateRatio field of the CUMULATIVE_RATE_RATIO TLV issued by PTP Ports on the Boundary Clock or Transparent Clock, to be received by the next downstream PTP Ports. This TLV shall be attached to (a) Sync messages if the message is transmitted or retransmitting from a one-step PTP Port, or (b) either to the Sync message or to the corresponding Follow_Up messages if the message is transmitted or retransmitted from a two-step PTP Port.

NOTE—An implementation can simplify the multiplication procedure, subject to any requirements of the applicable PTP Profile or the application. For example, if <neighborRateRatio> is sufficiently close to 1.0, addition of the quantity <neighborRateRatio> – 1.0 gives a result that is approximately equal to the result obtained by multiplying by <neighborRateRatio>. An implementation could choose to do this addition, rather than the multiplication by <neighborRateRatio>, if the result meets the accuracy requirements of the respective PTP Profile or application.

Table 123—CUMULATIVE_RATE_RATIO TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
scaledCumulativeRateRatio								4	4

16.10.3.1 tlvType

The value of tlvType shall be CUMULATIVE_RATE_RATIO.

16.10.3.2 lengthField

The value of the lengthField is 4.

16.10.3.3 scaledCumulativeRateRatio (Integer32)

The value of <cumulativeRateRatio> shall be the value computed as above. The scaledCumulativeRateRatio is computed as follows:

$\text{scaledCumulativeRateRatio} = (\text{<cumulativeRateRatio>} - 1) \times 2^{41}$ and truncated to the next smaller signed integer

NOTE—This scaling allows representation of fractional frequency offsets in the range $[-9.766 \times 10^{-4}, 9.766 \times 10^{-4}]$ and with a granularity of 2^{-41} .

16.10.4 Calculations based on the measured <meanLinkDelay>, <neighborRateRatio>, and cumulativeRateRatio field of the TLV

16.10.4.1 General

The specifications in 16.10.4 are predicated on the use of choice c) of 16.10.2 in the specification of the Timestamping Clock used in this option.

16.10.4.2 meanLinkDelay

Upon receipt of a Sync or Sync/Follow_Up pair, a PTP Instance shall multiply the <meanLinkDelay> value measured in 16.10.2 by the value of the <neighborRateRatio> pertinent to the ingress PTP Port as measured in 16.10.2, and then multiply the result by the value of the cumulativeRateRatio from the received CUMULATIVE_RATE_RATIO TLV (see 16.10.3). The result is the corrected value of the <meanLinkDelay>.

16.10.4.3 Ordinary Clocks that are not the Grandmaster Clock of the domain

The corrected value of the <meanLinkDelay> (see 16.10.4.2) shall be used in generating the time of the Local PTP Clock at this PTP Instance using received Sync and, in the two-step case, Follow_Up messages by minimizing the <offsetFromMaster> computed as specified in 11.2.

A PTP Profile may specify additional requirements on how minimization of <offsetFromMaster> is used to adjust the time of the Local PTP Clock to synchronize to its Parent Clock.

NOTE—Such requirements could consist of specifications on the use of PLLs, filters and other mechanisms.

16.10.4.4 Transparent Clocks and all Boundary Clocks that are not the Grandmaster PTP Instance of the domain

The <neighborRateRatio> and <cumulativeRateRatio> values specified in this subclause shall be those computed for the path used by the ingress Sync message at the PTP Instance.

NOTE—This path is specific to the domain.

The corrected value of the <meanLinkDelay> (see 16.10.4.2) shall be added to the correctionField of the Sync or Follow_Up message as appropriate.

When a Boundary Clock uses this option, it does not generate an egress Sync and, if appropriate, Follow_Up message as specified in 9.5.4, 9.5.5, 9.5.9, and 9.5.10, and the Boundary Clock is not required to synchronize a Local PTP Clock. Rather, it shall measure a <residenceTime> and transmit the ingress Sync and, if appropriate, Follow_Up messages, as soon as possible after receipt of the ingress Sync message.

The <residenceTime> corresponding to the time interval between the receipt of a Sync message on the ingress PTP Port and the transmission or retransmission of that Sync message on an egress PTP Port shall be multiplied by the value of the <neighborRateRatio> pertinent to the ingress PTP Port (see 16.10.2), and then multiplied by the value of the cumulativeRateRatio from the received Cumulative_Rate_Ratio TLV. The result is the corrected value of the <residenceTime>, which shall be added to the correctionField of the Sync or Follow_Up message as appropriate prior to transmission or retransmission on the egress PTP Port.

16.11 Slave Event Monitoring (optional)

16.11.1 General

When this option is both implemented (see 6.1), and enabled (see 16.11.2), then the option shall operate as specified in this subclause.

This option provides a mechanism for monitoring timing information in a PTP Port in the Slave state.

A PTP Network is principally focused on the one-way distribution of time to Slave PTP Instances. A Slave PTP Instance knows such parameters as the clock quality of the PTP Network's Grandmaster Clock, and may know the accumulated timing inaccuracy. External monitoring of signals tied to the Slave PTP Instance's Local PTP Clock (e.g., a 1 PPS signal) are typically employed to evaluate the Slave PTP Instance's performance in a PTP Network, but such signals can be subject to errors in delay calibrations, and in some products, implementation of the signals is impractical. Applications making use of precise time may have mechanisms to detect when one or more PTP Ports in the SLAVE state in the PTP Network have a large timing error; however, the slave event monitoring mechanism defined in this subclause provides an application-independent mechanism to ease the qualification, validation, and monitoring of the tracking of timing information by the PTP Port in the SLAVE state as delivered by the PTP Network.

A Slave PTP Instance may indicate the timestamps for certain PTP event messages on ingress or egress from the device's PTP Port in the SLAVE state for the domain and related data values at the time the PTP event message was received or sent.

A Slave PTP Instance may provide this data out-of-band in a vendor specific manner, or provide this data in-band on the PTP Network by transmitting SLAVE_RX_SYNC_TIMING_DATA TLVs and/or the SLAVE_RX_SYNC_COMPUTED_DATA TLVs and/or the SLAVE_TX_EVENT_TIMESTAMPS TLVs.

If a Slave PTP Instance supports the generation and transmission of a one or more Event Monitoring TLVs, then all of the following apply:

- The Slave PTP Instance shall support the optional Data Set values corresponding to the supported Event Monitoring TLVs.
- The members of these optional Data Sets shall be populated each time the defined values are determined by the implementation.
- The values of the TLVs shall be based on the accumulated values in these data sets.

The Slave PTP Instance may send a TLV containing information for each event message monitored, or the Slave PTP Instance may skip several event messages for each TLV. The Slave PTP Instance may include

information for several event messages in each TLV sent. The Slave PTP Instance may send multiple such TLVs attached to the same PTP message.

The Slave Event Monitoring mechanisms defined in this subclause do not specify the potential uses for this mechanism. To be effective, some use cases may require the PTP Port in the SLAVE state to be on a direct connection on a well-known communication path to a well-known PTP Port in the MASTER state.

An applicable PTP Profile that wishes to make use of this monitoring mechanism must specify which TLVs and sampling rates should be supported by a PTP Instance.

The Event Monitoring mechanism may be enabled or disabled by management configuration of the members of the slaveMonitoringPortDS.slaveEventMonitoringEnable, for example, by using the SLAVE_EVENT_MONITORING_ENABLE management TLV.

The use of slave event monitoring at special ports is outside the scope of this standard.

16.11.2 Slave Event Monitoring Enable

An implementation-specific enable control shall be maintained for the transmission of each TLV. If enabled for a TLV defined in this subclause, the Slave Event Monitoring mechanism shall be operational for that TLV. If disabled for a TLV defined in this subclause, the slave event monitoring mechanism shall be inactive for that TLV. The slave event monitoring mechanism shall be disabled by default unless otherwise specified in the applicable PTP Profile.

16.11.3 Transmission of PTP messages with Slave Event Monitoring TLVs

When the transmission of any Slave Event Monitoring TLV type is enabled, TLVs of the enabled TLV types shall be transmitted as part of a Signaling message unless otherwise specified in the applicable PTP Profile. The Signaling message shall be sent to a targetPortIdentity of all ones. The domainNumber in the transmitted Signaling message shall be the domainNumber of the related event message.

If not otherwise specified using a management mechanism, the Slave PTP Instance shall transmit each TLV, containing “N” sample values. Configuring the “Mth” event message to sample is determined by the slaveMonitoringPortDS parameters : slaveEventMonitoringRxSyncTimingEventMessageM, slaveEventMonitoringRxSyncComputedEventMessageM, and slaveEventMonitoringETxEventTimestampsEventMessageM. The value of “N,” how many samples to include in a TLV, is determined by the slaveMonitoringPortDS parameters slaveEventMonitoringEventsPerRxSyncTimingTLV, slaveEventMonitoringEventsPerRxSyncComputedTLV and slaveEventMonitoringEventsPerTxEventTimestamps TLV. Note that an implementation need not support all values. Refer to 16.11.6 for further details on these parameters. The periodicity of the TLV transmission is left to the implementation if not otherwise specified in the applicable PTP Profile. The Slave PTP Instance may transmit one or more Slave Event Monitoring TLVs attached to the same Signaling message.

The network addressing of the Signaling message carrying Slave Event Monitoring TLVs may be specified by an applicable PTP Profile such that a management Instance may collect and monitor data conveyed in the Slave Event Monitoring TLVs. Unless otherwise specified, the Signaling message shall use a multicast address, specified for PTP for the respective transport. If the Signaling message is sent to a unicast address, the Signaling message will be forwarded without alteration of the PTP defined fields per the transport layer forwarding rules of the PTP Network.

16.11.4 Ingress Event Monitor TLVs

The following Ingress Event Monitoring TLVs provide observability into the inner workings of a PTP Instance. A PTP Profile may require support for some or all of the following TLVs.

16.11.4.1 SLAVE_RX_SYNC_TIMING_DATA TLV specification

A Slave PTP Instance receiving Sync messages may utilize this TLV to provide observability of the instance's Local PTP Clock. For a specific Sync message, this TLV conveys not only the ingress timestamp of the Sync event message, but provides additional values transcribed from the monitored Sync of use for diagnostic purposes such as when assessing causes of timing error.

The SLAVE_RX_SYNC_TIMING_DATA TLV format shall be as specified in Table 124.

Table 124—SLAVE_RX_SYNC_TIMING_DATA TLV

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
syncSourcePortIdentity								10	4
sequenceId[1]								2	14
syncOriginTimestamp[1]								10	16
totalCorrectionField[1]								8	26
scaledCumulativeRateOffset[1]								4	34
syncEventIngressTimestamp[1]								10	38
....									
sequenceId[N]								2	14+(N-1)*34
syncOriginTimestamp[N]								10	16+(N-1)*34
totalCorrectionField[N]								8	26+(N-1)*34
scaledCumulativeRateOffset[N]								4	34+(N-1)*34
syncEventIngressTimestamp[N]								10	38+(N-1)*34

16.11.4.1.1 tlvType

The value of tlvType shall be SLAVE_RX_SYNC_TIMING_DATA.

16.11.4.1.2 lengthField

The value of the lengthField shall be 10+N*34, where N is the number of data element sets conveyed in the TLV.

16.11.4.1.3 syncSourcePortIdentity (PortIdentity)

The value of the syncSourcePortIdentity field shall be the value of the sourcePortIdentity of the received Sync message.

16.11.4.1.4 sequenceId[i] (UInteger16)

The value of the sequenceId[i] shall be the value of the sequenceId field of the *i*th Sync message.

16.11.4.1.5 syncOriginTimestamp[i] (Timestamp)

The value of syncOriginTimestamp[i] shall be the value of the syncEventEgressTimestamp value generated by the PTP Port sending the Sync message for the *i*th Sync message. If the twoStepFlag bit was set to TRUE the syncOriginTimestamp will be the preciseOriginTimestamp from the correlated Follow_Up message; otherwise, it will be the originTimestamp from the received Sync message.

16.11.4.1.6 totalCorrectionField[i] (TimeInterval)

The value of totalCorrectionField[i] shall be the aggregate value of the correctionField fields (i.e., of the Sync and/or Follow_Up message as appropriate) as received by the Slave PTP Instance. This value is used to determine the currentDS.offsetFromMaster for the *i*th Sync message. This value is therefore the sum of the correctionField of the received Sync message and the correctionField within the Follow_Up message if the received Sync message had the twoStepFlag bit set to TRUE.

16.11.4.1.7 scaledCumulativeRateOffset[i] (Integer32)

The value of scaledCumulativeRateOffset[i] shall be the value of the scaledCumulativeRateOffset value (see 16.10) received by the Slave PTP Instance related to the *i*th Sync message. For some profiles, such as IEEE Std 802.1AS, the cumulativeScaledRateOffset is conveyed in the Follow_Up Information TLV. This field is optional and may be sent as all zeros if the Slave PTP Instance does not utilize scaledCumulativeRateOffset.

16.11.4.1.8 syncEventIngressTimestamp[i] (Timestamp)

The value of syncEventIngressTimestamp[i] shall be the value of the syncEventIngressTimestamp acquired for the *i*th Sync message. This Timestamp is conveyed relative to the PTP timescale for the PTP Domain indicated in the domainNumber field of the PTP message carried in this TLV.

16.11.4.2 SLAVE_RX_SYNC_COMPUTED_DATA TLV specification

A Slave PTP Instance receiving Sync messages may utilize this TLV to provide observability of the Instance's computed currentDS.offsetFromMaster at the time the monitored Sync message was received, prior to any corrective updates.

This TLV may also convey the computed PTP Instance's <meanPathDelay> at the time the monitored Sync message was received. The TLV may also convey the <neighborRateRatio>, for those profiles that utilize it, at the time the monitored Sync message was received.

The data conveyed in this TLV is provided separately from the SLAVE_RX_SYNC_TIMING_DATA to allow for different sampling rates as may be configured or specified by a PTP Profile utilizing this TLV.

The SLAVE_RX_SYNC_COMPUTED_DATA TLV format shall be as specified in Table 125.

Table 125—SLAVE_RX_SYNC_COMPUTED_DATA TLV

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
sourcePortIdentity								10	4
computedFlags								1	14
reserved								1	15
sequenceId[1]								2	16
offsetFromMaster[1]								8	18
meanPathDelay[1]								8	26
scaledNeighborRateRatio[1]								4	34
....									
sequenceId[N]								2	16+(N-1)*2
offsetFromMaster[N]								8	18+(N-1)*2
meanPathDelay[N]								8	26+(N-1)*2
scaledNeighborRateRatio[N]								4	30+(N-1)*2

16.11.4.2.1 tlvType

The value of tlvType shall be SLAVE_RX_SYNC_COMPUTED_DATA.

16.11.4.2.2 lengthField

The value of the lengthField shall be $12+N*2$, where N is the number of data element sets conveyed in the TLV.

16.11.4.2.3 sourcePortIdentity (PortIdentity)

The value of the sourcePortIdentity field shall be the value of the sourcePortIdentity in the received Sync message.

16.11.4.2.4 computedFlags (Boolean[8])

The value of the computedFlags bits shall be as defined in Table 126:

Table 126—computedFlags field values

Bit	Description
0	scaledNeighborRateRatioValid
1	meanPathDelayValid
2	offsetFromMasterValid
3–7	reserved

16.11.4.2.4.1 offsetFromMasterValid flag

The value of the offsetFromMasterValid flag field shall indicate if the offsetFromMaster field in the SLAVE_RX_SYNC_COMPUTED_DATA TLV contains a valid value. If TRUE, then the offsetFromMaster field is valid for all “N” values conveyed in the TLV. If FALSE, one or more of the conveyed values is incorrect.

16.11.4.2.4.2 meanPathDelayValid flag

The value of the meanPathDelayValid flag field shall indicate if the meanPathDelay field in the SLAVE_RX_SYNC_COMPUTED_DATA TLV contains a valid value. If TRUE, then the meanPathDelayfield is valid for all “N” values conveyed in the TLV. If FALSE, one or more of the conveyed values is incorrect.

16.11.4.2.4.3 scaledNeighborRateRatioValid flag

The value of the scaledNeighborRateRatioValid flag field shall indicate if the scaledNeighborRateRatio field in the SLAVE_RX_SYNC_COMPUTED_DATA TLV contains a valid value. If TRUE, then the scaledNeighborRateRatio field is valid for all “N” values conveyed in the TLV. If FALSE, one or more of the conveyed values is incorrect.

16.11.4.2.5 sequenceId[i] (UInteger16)

The value of the sequenceId[i] shall be the value of the sequenceId field of the *i*th Sync message.

16.11.4.2.6 offsetFromMaster[i] (TimeInterval)

The *i*th value of offsetFromMaster[i] shall be the value specified in 11.2 at the time the Slave PTP Instance updates the currentDS.offsetFromMaster calculation for the *i*th Sync message. The contents in this field are known to be valid only when the value of the offsetFromMasterValid flag of the TLV is TRUE.

16.11.4.2.7 meanPathDelay[i] (TimeInterval)

The *i*th value of meanPathDelay[i] shall be the value specified in 11.3 and 11.4 at the time the Slave PTP Instance updates the currentDS.offsetFromMaster calculation for the *i*th Sync message. The contents in this field are valid only when the value of the meanPathDelayValid flag is TRUE.

16.11.4.2.8 scaledNeighborRateRatio[i] (Integer32)

The *i*th value of scaledNeighborRateRatio, that is, scaledNeighborRateRatio[i], shall be the <neighborRateRatio> value as specified by 16.6.3 when using the CMLDS of 16.6, otherwise by 16.10, at the time the Slave PTP Instance updates the currentDS.offsetFromMaster calculation for the *i*th Sync message. The value conveyed in this field shall be the <neighborRateRatio> scaled to a fixed point value; specifically, the scaledNeighborRateRatio value shall be the $(\text{<neighborRateRatio>} - 1.0) \times 2^{41}$, truncated to the next smaller signed integer. When provided, this field conveys the ratio of the frequency of the link partner's clock to the frequency of the Local Clock. The contents in this field are valid only when the value of the neighborRateRatioValid flag is TRUE.

16.11.5 Egress Event Monitor TLVs

16.11.5.1 SLAVE_TX_EVENT_TIMESTAMPS TLV specification

A PTP Instance transmitting event messages may utilize this TLV to provide observability of PTP Instance Time. For a selected event message type, the egress timestamp of the monitored PTP event message can be provided.

An example use of this TLV is to provide visibility of a Slave PTP Instance's delay request-response mechanism's t_3 value.

The SLAVE_TX_EVENT_TIMESTAMPS TLV format shall be as specified in Table 127.

Table 127—SLAVE_TX_EVENT_TIMESTAMPS TLV

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
sourcePortIdentity								10	4
reserved				eventMessageType				1	14
reserved								1	15
sequenceId[1]								2	16
eventEgressTimestamp [1]								10	18
....									
sequenceId[N]								2	16+(N-1)*12
eventEgressTimestamp[N]								10	18+(N-1)*12

16.11.5.1.1 tlvType

The value of tlvType shall be SLAVE_TX_EVENT_TIMESTAMPS.

16.11.5.1.2 lengthField

The value of the lengthField shall be $12+N*12$, where N is the number of data element sets conveyed in the TLV.

16.11.5.1.3 sourcePortIdentity (PortIdentity)

The value of the sourcePortIdentity field shall be the value of the sourcePortIdentity in the transmitted event message.

16.11.5.1.4 eventMessageType (Enumeration4)

The value of the eventMessageType shall be the messageType value of the event message monitored in this TLV. The coding of the enumeration is identical to that used in the messageType field of message headers (see 13.3.2.3).

16.11.5.1.5 sequenceId[i] (UInteger16)

The value of the sequenceId[i] shall be the value of the sequenceId field of the *i*th event message.

16.11.5.1.6 eventEgressTimestamp[i] (Timestamp)

The value of eventEgressTimestamp[i] shall be the value of the egress Timestamp acquired for the *i*th event message. This timestamp is conveyed in the timescale (either PTP or ARB, depending on the configuration) for the PTP Domain indicated in the PTP message's domainNumber field carrying this TLV.

NOTE—For an IEEE 802.11 port implementing Timing Measurement, this field is used to convey the t_3 value received by the Slave PTP Instance in a MLME.TIMINGMSMT.indication.

16.11.6 slaveMonitoringPortDS data set member specifications

16.11.6.1 General

The name and operational conformance, as defined in 8.1.6, of each slaveMonitoringPortDS member is summarized in Table 128.

Table 128—slaveMonitoringPortDS operational conformance

Name	OC	BC	E2E TC	P2P TC
slaveEventMonitoringEnable	management	management	management	managem
slaveEventMonitoringEventsPerRxSyncTimingTLV	management	management	management	managem
slaveEventMonitoringEventsPerRxSyncComputedTLV	management	management	management	managem
slaveEventMonitoringEventsPerTxEventTimestampsTLV	management	management	management	managem
slaveEventMonitoringTxEventType	management	management	management	managem
slaveEventMonitoringRxSyncTimingEventMessageM	management	management	management	manager
slaveEventMonitoringRxSyncComputedEventMessageM	management	management	management	manager
slaveEventMonitoringTxEventTimestampsEventMessageM	management	management	management	manager

This data set represents an optional mechanism for the control of slave monitored timestamps that occur in the PTP Instance.

16.11.6.2 slaveMonitoringPortDS.slaveEventMonitoringEnable

The value of slaveMonitoringPortDS.slaveEventMonitoringEnable indicates whether the data for the slave event monitoring TLVs of 16.11 are computed, and whether the data are transmitted by the slave. The data type shall be Boolean [16]. The values are defined in Table 129.

Table 129—slaveEventMonitoringEnable values

Index	Description
0	SLAVE RX SYNC TIMING DATA generation is active if TRUE, otherwise the generation is inactive
1	SLAVE RX SYNC COMPUTED DATA generation is active if TRUE, otherwise the generation is inactive
2	SLAVE TX EVENT TIMESTAMPS generation is active if TRUE, otherwise the generation is inactive
3–15	reserved

The specification initialization value (see 8.1.3.4) shall be set to FALSE. The classification is configurable.

16.11.6.3 slaveMonitoringPortDS.slaveEventMonitoringEventsPerRxSyncTimingTLV

The value of slaveMonitoringPortDS.slaveEventMonitoringEventsPerRxSyncTimingTLV indicates the number of events to report per SLAVE_RX_SYNC_TIMING_DATA TLV. The data type shall be UInteger8. The classification is configurable. This value is used only for construction of transmitted TLVs when TLV transmission is enabled. An implementation need not support all values. Any attempt to configure this value to an unsupported value shall not result in the value changing. The specification initialization value (see 8.1.3.4) shall be 0.

16.11.6.4 slaveMonitoringPortDS.slaveEventMonitoringEventsPerRxSyncComputedTLV

The value of slaveMonitoringPortDS.slaveEventMonitoringEventsPerRxSyncComputedTLV indicates the number of events to report per SLAVE_RX_SYNC_COMPUTED_DATA TLV. The data type shall be UInteger8. The specification initialization value (see 8.1.3.4) shall be 0. The classification is configurable. This value is used only for construction of transmitted TLVs when TLV transmission is enabled. An implementation need not support all values. Any attempt to configure this value to an unsupported value shall not result in the value changing.

16.11.6.5 slaveMonitoringPortDS.slaveEventMonitoringEventsPerTxEventTimestampsTLV

The value of slaveMonitoringPortDS.slaveEventMonitoringEventsPerTxEventTimestampsTLV indicates the number of events to report per SLAVE_TX_EVENT_TIMESTAMPS_DATA TLV. The data type shall be UInteger8. The specification initialization value (see 8.1.3.4) shall be 0. The classification is configurable. This value is used only for construction of transmitted TLVs when TLV transmission is enabled. An implementation need not support all values. Any attempt to configure this value to an unsupported value shall not result in the value changing.

16.11.6.6 slaveMonitoringPortDS.slaveEventMonitoringTxEventType

The value of slaveMonitoringPortDS.slaveEventMonitoringTxEventType indicates the event message type selected for the egress event monitoring. The data type shall be UInteger8, where the four low-order bits are defined to correspond to enumeration of Table 36 (Values of messageType field) and shall default to 1 (Delay_Req). The classification is configurable. An implementation need not support all values. Any attempt to configure this value to an unsupported value shall not result in the value changing.

16.11.6.7 slaveMonitoringPortDS.slaveEventMonitoringRxSyncTimingEventMessageM

The value of slaveMonitoringPortDS.slaveEventMonitoringRxSyncTimingEventMessageM is the value M, where M indicates that every Mth event message is selected for monitoring in the SLAVE_RX_SYNC_TIMING_DATA TLV. For example, if the value of M is 4, every fourth event message is selected for monitoring in the TLV. The data type shall be UInteger8. The default value shall be 1. The classification is configurable. Any attempt to configure this value to an unsupported value shall result in the value not changing.

16.11.6.8 slaveMonitoringPortDS.slaveEventMonitoringRxSyncComputedEventMessageM

The value of slaveMonitoringPortDS.slaveEventMonitoringRxSyncComputedEventMessageM is the value M, where M indicates that every Mth event message is selected for monitoring in the SLAVE_RX_SYNC_COMPUTED_DATA TLV. For example, if the value of M is 4, every fourth event message is selected for monitoring in the TLV. The data type shall be UInteger8. The default value shall be 1. The classification is configurable. Any attempt to configure this value to an unsupported value shall result in the value not changing.

16.11.6.9 slaveMonitoringPortDS.slaveEventMonitoringTxEventTimestampsEventMessageM

The value of slaveMonitoringPortDS.slaveEventMonitoringTxEventTimestampsEventMessageM is the value M, where M indicates that every Mth event message is selected for monitoring in the SLAVE_TX_EVENT_TIMESTAMP TLV. For example, if the value of M is 4, every fourth event message is selected for monitoring in the TLV. The data type shall be UInteger8. The default value shall be 1. The classification is configurable. Any attempt to configure this value to an unsupported value shall result in the value not changing.

16.12 Enhanced synchronization accuracy metrics (optional)

16.12.1 General

When this option is implemented (see 6.1), then the option shall operate as specified in this subclause.

This subclause specifies a mechanism, using a TLV, for propagating estimates of various inaccuracy components affecting the overall expected PTP Instance Time accuracy. The metrics will be updated and available for utilization at the various points along the PTP timing chain, from the Grandmaster Instance, up to a leaf PTP Instance in the synchronization tree. Each implementing node along the timing path updates the relevant metrics based on its contribution to the expected degradation in PTP Instance Time accuracy due to various induced timing error components.

The mandatory parts of this standard include, within the clockQuality, basic metrics depicting the accuracy of time distributed by the GM. When Boundary Clocks are part of the timing chain, the number of those traversed at various points in the chain is depicted by the stepsRemoved metric within the Announce message. The latter however does not convey any effects of Transparent Clocks in the path and does not provide the ability to effectively generate a quantitative metric for PTP Instance Time accuracy downstream from the Grandmaster PTP Instance in a generic manner (especially when the PTP Network includes both Boundary Clocks and Transparent Clocks).

Various applications need, or can benefit from, more precise quantitative information regarding the current estimated PTP Instance Time accuracy available within the various PTP Instances in the network. These can be used, for example, to verify that timing performance is adequate for the applications needs or, possibly, to adapt application behavior to the accuracy currently available (e.g., by adaptation of operational margins).

In assessing the expected accuracy, the use of such a metric can be valuable in determining the values to be advertised when a Slave PTP Instance serves as a timing source to a different PTP Network (see O.4.3).

The metrics included in the ENHANCED_ACCURACY_METRICS TLV might also be used within PTP Profiles to specify the establishment of the PTP network topology. Such a PTP Profile might, for example, include an alternate BMCA that uses these metrics to create a topology more optimized for the performance requirements of its applications. Such a PTP Profile must mandate support for this option by all relevant PTP Instances.

To support the varying needs within these aforementioned applications and other possible ones, this option supports dividing the various components affecting the achieved accuracy into several contributing component categories. The sum of the contribution of components from these categories supplies an overall estimate for the expected accuracy. The categories are as follows:

- Grandmaster inaccuracy
- Transient path inaccuracy

- Dynamic path inaccuracy
- Static path inaccuracy (separated into contribution within a PTP Instance and on the preceding medium where relevant)

The categories providing information on the timing degradation in the timing transfer from the Grandmaster (i.e., Transient, Dynamic and Static) are termed “distribution components” in this subclause.

In some cases, the worst-case performance under normal operating conditions is of interest, whereas in others, a metric indicating the typical range of these error components is of interest. The TLV contains two types of metrics for each of these components: maximum and variance.

Grandmaster Clocks, Boundary Clocks, and Transparent Clocks each update their contribution into the appropriate members of the ENHANCED_ACCURACY_METRICS TLV as specified below. This TLV is an optional mechanism that should be supported by all PTP Instances participating within the domain as only then will all contributions be accounted for at the leaf PTP Instances of the topology.

If `enhancedSynchronizationAccuracyMetricsDS.enable` is TRUE within a Grandmaster PTP Instance supporting this option, the ENHANCED_ACCURACY_METRICS TLV shall be appended to all outgoing Announce messages transmitted by such a PTP Instance. A Grandmaster PTP Instance shall provide in the relevant Grandmaster metrics defined below (`maxGmInaccuracy` `varGmInaccuracy`) estimates regarding the accuracy of the timing it is distributing. It shall initialize all the distribution components, the `bcHopCount` and the `tcHopCount`, to zero in the transmitted TLV.

The TLV shall be a propagating TLV (see 14.2.2.2), and thus will be propagated down the topology by all Boundary Clocks receiving it (regardless of whether they support this option).

If `enhancedSynchronizationAccuracyMetricsDS.enable` is configured TRUE within a Boundary Clock supporting this option, it shall update its contributions to all the appropriate metrics in the propagated ENHANCED_ACCURACY_METRICS TLV attached to an incoming Announce message, prior to propagating it: that is, add its own contribution to the distribution components. It shall also increment the received value of `bcHopCount` by one, and use this incremented value in the propagated TLV. A Boundary Clock shall not update any other fields within the ENHANCED_ACCURACY_METRICS TLV.

If `enhancedSynchronizationAccuracyMetricsDS.enable` is configured TRUE within a Transparent Clock supporting this option, it shall update its contributions to all the appropriate metrics in the retransmitted ENHANCED_ACCURACY_METRICS TLV attached to an incoming Announce message, prior to retransmitting it: that is, add its own contribution to the distribution components. It shall also increment the received value of `tcHopCount` by one, and use this incremented value in the retransmitted TLV. A Transparent Clock shall not update any other fields within the ENHANCED_ACCURACY_METRICS TLV.

Both these types of redistributing PTP Instances (i.e., Boundary Clocks and Transparent Clocks) shall update the path metrics based upon both their own internal contributions and the contributions introduced along the timing link from the preceding PTP Instance (see Figure 46).

NOTE 1—The `bcHopCount` field can be used to determine whether all BC’s along the announce message path update the TLV by comparing it to the `stepsRemoved` field of the Announce message.

NOTE 2—Transparent Clocks supporting this option will update the metrics and increase the `tcHopCount` field. Transparent Clocks not supporting this option will retransmit the entire message to which the TLV is appended, including the appended TLV, as done for all messages. There is no protocol mechanism enabling the determination of whether all Transparent Clocks along the path support this option. A profile could require Transparent Clock support for this option.

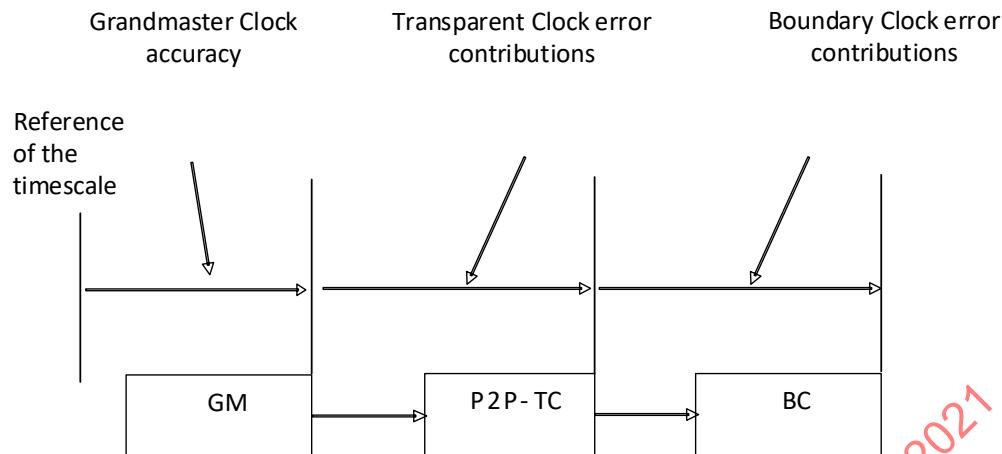


Figure 46—Contribution of the various PTP Instances to metrics in the TLV

The TLV includes the following two groups of components:

- **GM component:** A component depicting the timing performance supplied by the Grandmaster PTP Instance (as compared with the timescale being distributed by it). (The component will be quantified using the metrics defined below.)
- **Distribution components:** Components depicting the reduction in accuracy expected along the timing path during the normal operation of the PTP Network, including transitional contributors that can occur during network transitions (e.g., caused by a change in the topology, or during the initial start-up of the network) (These component will be quantified using the metrics defined below, and accumulated in the manner defined below.)

The different types of distribution components are illustrated by Figure 47, depicting various possible contributors to the PTP Instance Time accuracy.

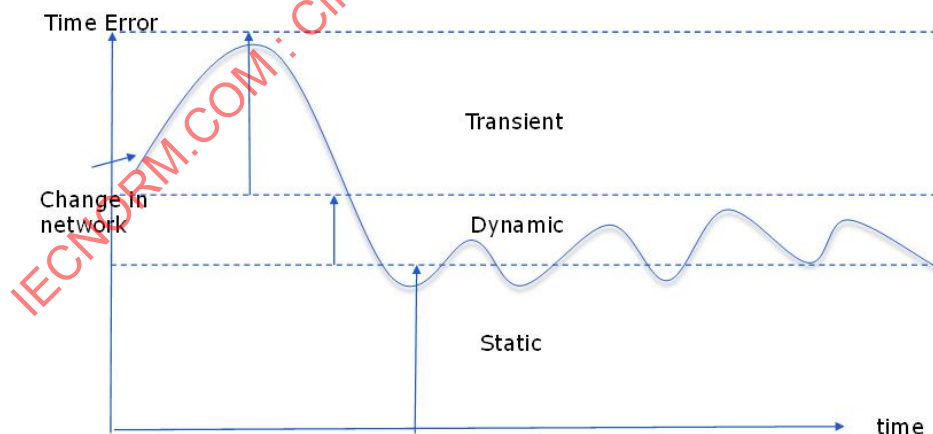


Figure 47—Contributions to degradation in time distribution

For the distribution components, the metrics shall be based on all knowledge the PTP Instance has regarding the current deviations it (and the link connecting it to the upstream instance) might be adding to the timescale being distributed by the upstream PTP instance. Since the different components might have different significance in context of the application (e.g., the dynamic component could possibly be filtered

and could be considered of a different significance compared to the static component), the metrics enable the quantification of the division between the defined components. The sum of all contributors shall quantify all possible contributors. If an error contributor is not known to be a transient or dynamic component (per their definitions below), its contribution should be accounted for within the static component.

The distribution components are divided into the following two subgroups:

- a) **Transient component:** It quantifies an additional time error currently estimated to be induced (beyond that expected in steady state) due to transitory convergence effects (e.g., due to a reaction to a PTP network rearrangement) dependent on the mechanisms by which the PTP Instance maintains its Local PTP Clock or Local Clock as appropriate. The absolute size of this component must decay to zero over time during steady state operation.

NOTE 3—Providing the metrics for the transient component in a Boundary Clock will, in general, involve estimation of the current deviation of `currentDS.offsetFromMaster` measurements from such measurement in steady state operation.

- b) **Steady state components:** These components quantify the remaining contributors affecting PTP Instance Time accuracy after all transient contributors (e.g., due to PTP network rearrangement) have become negligible in their effect. These are as follows:
 - 1) **Static component:** It quantifies the mean of the steady state time error under the current operating conditions of the PTP Instance.
 - 2) **Dynamic component:** It quantifies the temporal deviations of time error from its mean. The value for the dynamic component shall either include all the effects of relevant contributors or be an estimate accounting for the possible reduction in the high-frequency portion of this component as specified in the applicable PTP Profile (e.g., due to filtering in the PTP instance).

NOTE 4—The values to be supplied for the metrics for the steady state components will, in general, involve taking into account various aspects in the design of the implementation (e.g., variations expected due to possible variations in delay of components in use, and possible variations in the performance of the Local Clock utilized within the PTP Instance). It is likely that, while direct measurements can be used to verify the validity of the supplied estimates, they will not be a practical approach to ascertain the required values (e.g., as the estimates cover the expected variations among identically implemented PTP Instances over the expected ensemble of all relevant variations, e.g., those due to variations within production and the various deployment conditions).

The static component is further divided into the following two subcomponents that accumulate into its overall contribution:

- **staticInstance:** It quantifies the mean of the steady state time error under the current operating conditions of the PTP Instance, contributed by errors internal to the PTP Instance.
- **staticMedium:** It quantifies the mean of the steady state time error under the current operating conditions, contributed by the medium connecting the PTP Instance to the upstream PTP Instance.

The minimal observation window to be used to verify, based on an empirical time error measurement, that the PTP Instance is providing conservative estimates of the dynamic component (i.e., overestimating the error) should be supplied as part of the PTP Instance specifications.

For each component, the following two metrics shall be supplied:

- **Var:** A conservative (over-)estimate of the error contribution of the component.
- **Max:** A conservative (over-)estimate of the maximum of the absolute value of the error contribution of the component.

The current PTP Instance's own contributions (labeled "local(X)" in the following equation) shall be accumulated with values of the metrics within the received `ENHANCED_ACCURACY_METRICS` TLV

(labeled “received (X)” in the following equation), to generate the values to be included within the ENHANCED_ACCURACY_METRICS TLV transmitted by the PTP Instance (labeled “transmitted (X)” in the following equation). The accumulation shall be via addition of the PTP Instance’s own contribution to the received values:

$$\text{transmitted}(X) = \text{received}(X) + \text{local}(X)$$

where X is any of the metrics: maxTransientInaccuracy, varTransientInaccuracy, maxDynamicInaccuracy, varDynamicInaccuracy, maxStaticInstanceInaccuracy, varStaticInstanceInaccuracy, maxStaticMediumInaccuracy, varStaticMediumInaccuracy.

The max metrics shall be represented using the TimeInterval data type.

The var metrics shall be represented using the Float64 data type (see 5.2) in units of seconds squared (s^2). For example, if the expected value for the square root of the variance is 1 ns, then the value of the var estimate is represented as 10^{-18} (i.e., $10^{-9} \times 10^{-9}$) using the Float64 data type.

NOTE 5—Although the data type enables negative values, the definitions of the metrics generate only non-negative values.

16.12.2 ENHANCED_ACCURACY_METRICS TLV specification

The ENHANCED_ACCURACY_METRICS TLV format shall be as specified in Table 130.

Table 130—ENHANCED_ACCURACY_METRICS TLV format

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
bcHopCount								1	4
tcHopCount								1	5
Reserved								2	6
maxGmInaccuracy								8	8
varGmInaccuracy								8	16
maxTransientInaccuracy								8	24
varTransientInaccuracy								8	32
maxDynamicInaccuracy								8	40
varDynamicInaccuracy								8	48
maxStaticInstanceInaccuracy								8	56
varStaticInstanceInaccuracy								8	64
maxStaticMediumInaccuracy								8	72
varStaticMediumInaccuracy								8	80

16.12.2.1 tlvType

The value of tlvType shall be ENHANCED_ACCURACY_METRICS.

16.12.2.2 lengthField

The value of the lengthField is 84.

16.12.2.3 bcHopCount (UInteger8)

The value of the bcHopCount field is the number of Boundary Clocks that updated the content of this TLV.

16.12.2.4 tcHopCount (UInteger8)

The value of the tcHopCount field is the number of Transparent Clocks that updated the content of this TLV.

16.12.2.5 maxGmInaccuracy (TimeInterval)

The value of maxGmInaccuracy shall be the value of the Max metric (see 16.12.1) for the Grandmaster PTP Instance expressed in nanoseconds multiplied by 2^{16} .

16.12.2.6 varGmInaccuracy (Float64)

The value of varGmInaccuracy shall be the value of the Var metric (see 16.12.1) for the Grandmaster PTP Instance expressed in $[s^2]$.

16.12.2.7 maxTransientInaccuracy (TimeInterval)

The value of maxTransientInaccuracy shall be the value of the Max metric for the transient component expressed in nanoseconds multiplied by 2^{16} .

16.12.2.8 varTransientInaccuracy (Float64)

The value of varTransientInaccuracy shall be the value of the Var metric for the transient component expressed in $[s^2]$.

16.12.2.9 maxDynamicInaccuracy (TimeInterval)

The value of maxDynamicInaccuracy shall be the value of the Max metric for the dynamic component expressed in nanoseconds multiplied by 2^{16} .

16.12.2.10 varDynamicInaccuracy (Float64)

The value of varDynamicInaccuracy shall be the value of the Var metric for the dynamic component expressed in $[s^2]$.

16.12.2.11 maxStaticInstanceInaccuracy (TimeInterval)

The value of maxStaticInstanceInaccuracy shall be the value of the Max metric for the staticInstance expressed in nanoseconds multiplied by 2^{16} .

16.12.2.12 varStaticInstanceInaccuracy (Float64)

The value of varStaticInstanceInaccuracy shall be the value of the Var metric for the staticInstance expressed in [s²].

16.12.2.13 maxStaticMediumInaccuracy (TimeInterval)

The value of maxStaticMediumInaccuracy shall be the value of the Max metric for the staticMedium expressed in nanoseconds multiplied by 2¹⁶.

16.12.2.14 varStaticMediumInaccuracy (Float64)

The value of varStaticMediumInaccuracy shall be the value of the Var metric for the staticMedium expressed in [s²].

16.13 Message Length Extension (optional)

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile, then the option shall operate as specified in this subclause.

This option provides a mechanism for ensuring that PTP event messages traveling over the same network paths have the same length.

The presence of non-PTP network elements increases the probability that measured path delays for PTP event messages will depend on the length of these messages. This is particularly true if store and forward technology is used in such devices. The detrimental effects this generates (e.g., asymmetry) can often be minimized by ensuring that the appropriate PTP messages have the same length. The specifications of 9.5.9.3 provide for adjusting the lengths of PTP messages.

A PTP Profile can specify that this option is required. The PTP Profile must then specify the messageLength for each PTP message type allowed in the PTP profile. When this option is required by the applicable PTP Profile, the specifications of 9.5.9.3 are in effect. If the PTP Profile specifies this option but fails to specify the messageLength requirements, then by default the following messageLength specifications shall hold:

- a) If no other TLVs are appended to the PTP event messages used to measure path delay, then:
 - 1) When the delay request-response mechanism is used, the Sync and Delay_Req messages shall have the length specified in Clause 13, that is, no adjustment of messageLength is made.
 - 2) When the peer-to-peer delay mechanism is used, a PAD TLV of length 10 shall be appended to Sync messages. No adjustment of Pdelay_Req or Pdelay_Resp message lengths shall be made. The result is that the messageLength attribute for Sync, Pdelay_Req, and Pdelay_Resp messages have the same value.
- b) If there are other TLVs appended to the PTP event messages used to measure path delay, then:
 - 1) When the delay request-response mechanism is used, a PAD TLV of the required length shall be appended to the shorter of the Sync and Delay_Req messages such that the resulting messageLengths of the Sync and Delay_Req messages have the same value.
 - 2) When the peer-to-peer delay mechanism is used, PAD TLVs of the appropriate lengths shall be appended to Sync, Pdelay_Req and Pdelay_Resp messages as needed such that the resulting messageLengths of the Sync, Pdelay_Req and Pdelay_Resp messages have the same value.

NOTE 1—Message length alignment increases the likelihood that for non-PTP network devices that have equal ingress and egress line rates:

- The path delays calculated for Sync and Delay_Req messages when using the delay request-response mechanism are equal, and
- The PTP link delays calculated for the Sync, Pdelay_Req, and Pdelay_Resp messages when using the peer-to-peer mechanism are equal

However, message length alignment in general does not necessarily result in equal delays in the forward and reverse directions particularly when the ingress and egress line rates differ. This problem is most pronounced in non-PTP network elements using store-and-forward technology.

NOTE 2—When using peer-to-peer delay mechanism, this option can be useful when the peer messages traverse non-PTP nodes. In this case, the use of this option ensures that the Sync, Pdelay_Req, and Pdelay_Resp messages are the same length.

16.14 PTP integrated security mechanism (optional)

16.14.1 General

This option specifies a security extension to PTP. The PTP security extension provides source authentication, message integrity, and replay attack protection for PTP messages within a PTP domain. See Annex P for a more comprehensive discussion on security. The PTP security extension is realized by the following two basic mechanisms:

- Definition of the AUTHENTICATION TLV, allowing for source authentication and message integrity. The AUTHENTICATION TLV carries all necessary information to enable security processing for the sender and the receiver. Note that the AUTHENTICATION TLV has been defined to allow the application of a symmetric key to protect the PTP message. Depending on the associated key management, the AUTHENTICATION TLV supports two different verification approaches: immediate security processing and delayed security processing (see below). The AUTHENTICATION TLV is applicable for both unicast and multicast PTP message transmission.
- Associated key management, which allows for the secure distribution of all necessary security parameters required to construct or verify the AUTHENTICATION TLV. Note that the specific realization of the key management is left open here, as it is a precondition for the AUTHENTICATION TLV. Hence, manual and automated key management may be supported. Annex P provides examples for automated key management including the distribution of keys to be used in group communication for either immediate or delayed security processing.

The verification approach for the security information provided by the AUTHENTICATION TLV relates to the utilized key management scheme. As stated, the following two approaches are distinguished:

- Immediate security processing enables the processing of the AUTHENTICATION TLV before the content of the PTP message is further processed. This is supported by a key management, which typically establishes shared secrets either pair wise (for unicast communication) or group based (for multicast communication). Due to the nature of immediate processing, the security parameters, especially the shared secret, are such that they are known to the involved entities before packet processing. For example, this allows the change of the mutable fields like the correctionField by a Transparent Clock.
- Delayed security processing enables the delayed distribution of all security parameters or a subset of them (at least the secret key). This approach allows the application of a shared secret for source authentication, as it is disclosed securely from the entity generating the ICV when the security parameters are no longer used for generation and only used for validation in the PTP Instances. This method requires the storage of messages on the receiver side until the required security parameters have been provided for a posteriori verification of the stored PTP messages. This approach does not support the protection of mutable fields in the messages. Therefore, if there are

mutable fields in the PTP message, like the correctionField, they must be treated as having a value of 0 during the ICV calculation.

The selection of the verification approach can influence the support of mutable fields. The term “mutable” is used here to indicate that PTP Instances can make changes to the PTP message; for example, a Transparent Clock provides or updates the correctionField value. Delayed security processing at a receiving PTP client will not permit the verification of mutable fields. The mutable field in the context of delayed security processing of this specification is the correctionField. The handling of mutable fields for delayed security processing other than the correctionField is left for future editions to specify as part of the Security Policy Database (SPD) (see 3.1.69).

Annex P provides more details related to the association of automated key management for both verification schemes.

16.14.1.1 PTP message Enhancement

The PTP security extension is intended to provide source authentication and message integrity for PTP messages. When the PTP security extension is used, PTP messages appear as shown in Figure 48. The mechanism consists of the following two parts:

- AUTHENTICATION TLV carrying the information required for the cryptographic verification
- Rules for AUTHENTICATION TLV processing

The approach allows for immediate processing as well as for delayed processing. As stated, it is assumed that a key management protocol negotiates the security parameter for the Security Association (SA; see 3.1.66). As shown in Figure 48, the integrity protection enabled through the inner AUTHENTICATION TLV covers all preceding fields in the PTP message, starting with and including the PTP header. The part of the PTP message that is integrity protected is shown by the dashed line. If there are TLVs to be handled by intermediate PTP instances that are to be excluded from integrity protection, they shall be placed after the AUTHENTICATION TLV.

Two security processing schemes are supported, immediate and delayed security processing. There are use cases, in which both approaches can be used, leading to two AUTHENTICATION TLVs in a PTP message. An example use case would be a Boundary Clock that is also the Grandmaster PTP Instance using the delayed security processing approach to protect the entire PTP message excluding the correctionField. If there are Transparent Clocks between the Grandmaster PTP Instance and a downstream PTP Boundary Clock or Ordinary Clock, these Transparent Clocks may update the correctionField within the PTP message. To provide protection for the correction field, the Transparent Clock may use the immediate security scheme to protect the PTP message including the correctionField. The possible utilization of both delayed and immediate authentication for the same message is illustrated in Figure 48 by the inclusion of an outer security TLV including an ICV calculated over the message part covered by the dotted line.

Note that usage of multiple AUTHENTICATION TLVs is only permitted if the PTP message header fields, which are protected by both AUTHENTICATION TLVs, are unchanged by the intermediate PTP Instances.

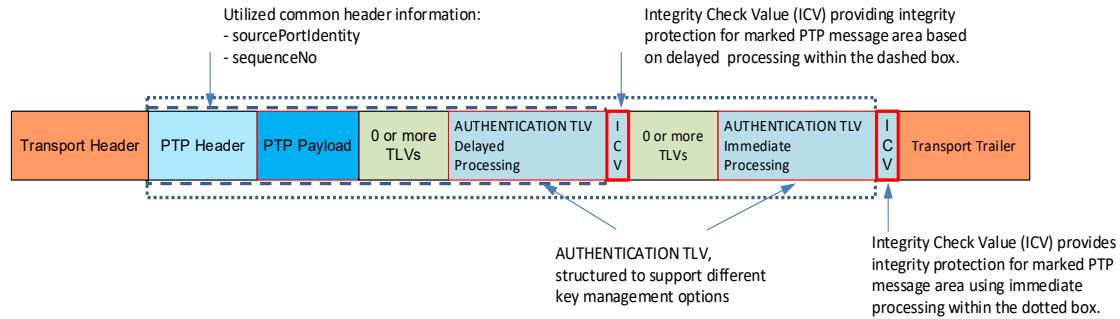


Figure 48—PTP message with security enhancements and used common header parts

The components of the AUTHENTICATION TLV and their processing are explained in the following subclauses. Also addressed are requirements to the supporting key management.

16.14.2 Requirements of the associated key management

As the security extension ensures the integrity of the time information, it strongly depends on the authenticity of the PTP Instance. It is therefore required that the authenticity of the PTP Instance be verified as part of the key management.

As stated, the key management to setup the security parameter for AUTHENTICATION TLV processing are not defined as part of this document. Nevertheless, there are some requirements for a minimum set of parameters that the key management must provide. These requirements are listed below.

The key management shall provide at least the following information to enable the calculation of the ICV:

- Security Parameter Pointer (SPP): Identifies the SA negotiated by the key management protocol (see 3.1.68).
- IntegrityAlgTyp: Identifies integrity algorithm type used to compute the ICV associated with each SPP. Note that the algorithm type is defined by an OID. The OID for the algorithm recommended in this specification is defined in FIPS PUB 198-1. Other OIDs defined by other standardization organizations are registered at IANA²¹ and may be used if other algorithms are to be applied.
- icvLength: Indicates the length of the calculated ICV (depends on the value of “K” in Table 131).
- Key: A symmetric key to be used in conjunction with the selected algorithm.
- keyLength: Indicates the length of the disclosed key (depends on the value of “D” in Table 131).
- Indication for use of sequenceNo and the desired length of the sequenceNo field (depends on the value of “S” in Table 131).
- Indication of the sequenceID window for anti-replay (based on the sequenceID in the PTP message header).
- Immediate Security: A Boolean indicating whether the SA is for immediate security or delayed security.
- Indication for use of RES and the desired length of the RES field (depends on the value of “R” in Table 131).

The information negotiated by the key management shall be stored in the Security Association Database (SAD; see 3.1.67). The implementation of and interaction with the SAD and SPD is implementation specific and outside the scope of this standard.

²¹ Internet Assigned Numbers Authority (<https://www.iana.org/>).

A PTP profile should define the key management or interaction with an existing key management for the parameters stated above, to enable PTP Instances to interoperate using the AUTHENTICATION TLV. Annex P provides examples for key management schemes for immediate and delayed security processing. The selection of the appropriate key management is typically a system integrator decision. The integrity and confidentiality of key management data and an appropriate configuration of the key management are critical to enabling the authentication scheme to meet its intended purpose.

NOTE—Preferably, either the key management or the security policy provides a list of mutable fields.

16.14.3 Definition of the AUTHENTICATION TLV

The AUTHENTICATION TLV defines all necessary fields to enable security processing for the sender and the receiver. To ensure minimal transport overhead, the AUTHENTICATION TLV is used in conjunction with a SAD, which stores all SA-specific information. Based on the specifications of Clause 14, the structure of the AUTHENTICATION TLV shall be as shown in Figure 49.

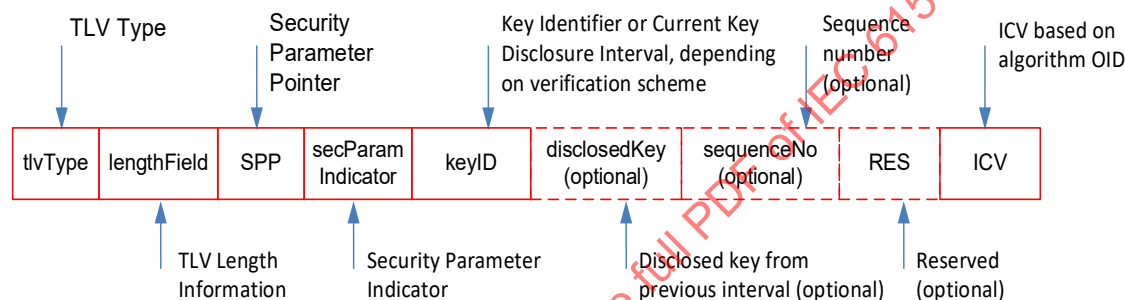


Figure 49—AUTHENTICATION TLV used for direct and delayed message security processing

Note that the AUTHENTICATION TLV has been defined to allow the application of a symmetric key to cryptographically protect the integrity of PTP messages. It supports immediate security processing, for example, by using a (group) key distributed upfront (before the PTP message communication is initiated). It also supports delayed security processing by providing an optional field for the disclosed key. The selection of the security scheme is a matter of the security policy of the user.

The AUTHENTICATION TLV may be appended to all PTP messages according to the security policy. An authentication TLV will be suffixed to the PTP message if required by the applicable security policy (see 16.14.4.1).

The field values of the AUTHENTICATION TLV shall be determined as defined in Table 131. Reserved fields are required to be set to zero on transmit and ignored on receipt (see 4.2.7).

Table 131—AUTHENTICATION TLV

Bits								Octets	TLV offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
SPP								1	4
secParamIndicator								1	5
keyID								4	6
disclosedKey (optional)								D	10
sequenceNo (optional)								S	10+D
RES (optional)								R	10+D+S
ICV								K	10+D+S+R

The values of D, S, R, and K shall be even values, and each value shall represent the number of octets in the field.

NOTE—The requirement that D, S, R, and K are even values guarantees that any combination of options will also result in an even number of octets (see 5.3.8).

The following subclauses define the parameters contained in the AUTHENTICATION TLV.

16.14.3.1 tlvType

The value of tlvType shall be AUTHENTICATION.

16.14.3.2 lengthField

The value of the lengthField must indicate the length of the overall AUTHENTICATION TLV payload (see 5.3.8). As the payload may contain optional fields and variable length fields, the processing of the optional field depends on the value of the lengthField. The existence and length of fields directly relates to the negotiated security parameters as part of the key management.

The value of lengthField shall be $6+D+S+R+K$.

16.14.3.3 SPP (UInteger8)

The value of the SPP shall be the value of the security parameter pointer for a specific security association. It shall be provided by the key management.

NOTE—The SPP defined in this subclause is not identical to the SPI defined in IETF RFC 4302. While the SPP is part of the attributes needed to identify the relevant SA, it can remain constant while the key associated with the SA is updated (with the key currently applicable for an SA further identified by the keyID).

An SPP enables querying the SAD to obtain the SA relevant for the processing of a AUTHENTICATION TLV. The SAD holds all specific parameters needed to calculate or recalculate the ICV field in the security TLV, while the SPD holds the policy parameters related with a specific session, for example, which messages should be secured (relevant when mixed message processing, that is, both secure processing and nonsecure processing, is supported).

The negotiation of parameters stored in the SAD and the associated security policy is typically done by the associated key management mechanism. For this edition, the definition of the security policy is out of scope.

The SAD should also record information differentiating individual PTP messages needed to support replay detection. This information depends on the negotiated security services defined by the SA parameters but must include the following:

- The sequenceID of the common header
- The messageType field of the common header

16.14.3.4 secParamIndicator (Boolean[8])

The security parameter indicator shall indicate the existence of optional fields in the AUTHENTICATION TLV as shown in Table 132.

Table 132—secParamIndicator

Bits							
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	DP	SP	RP

secParamIndicator indicates the presence or absence of the three optional fields as shown in Table 132. In Table 132, DP refers to the disclosedKey, SP to the sequenceNo, and RP to the RES optional fields. A value of FALSE for DP, SP, or RP shall indicate the absence of the corresponding optional field in the AUTHENTICATION TLV. A value of TRUE for DP, SP, or RP shall indicate the presence of the corresponding optional field in the AUTHENTICATION TLV. All bits indicated with a value Reserved in Table 132 are reserved for future use (see 4.2.7).

16.14.3.5 keyID (Octet [4])

The meaning of the keyID field depends on the utilized key management. In the following cases:

- **Immediate security processing:** The keyID defines the value of the currently used key. The keyID is to be provided by the key management protocol and typically stored inside the SAD. The explicit signaling of the keyID addresses key update cases in which an overlap of valid keys may occur.
- **Delayed security processing:** The keyID shall indicate the currently used key (either directly or indirectly depending on the delayed security scheme applied). The key shall be disclosed later. The earliest possible time of the disclosure is a function of the of the key disclosure delay, where the function depends on the delayed security scheme applied. The key disclosure delay shall be provided by the key management protocol and stored inside the SAD. Received PTP messages have to be buffered until their associated keys are disclosed for later authentication.

Note that in contrast to the SPP, the keyID can change during the lifetime of a security association. A change of the keyID field is the result of a key update.

16.14.3.6 disclosedKey (Octet[D])

This field is optional. If present, it shall have a data type of UInteger with a length of D octets. D must be an even integer.

This field is only applicable if the delayed security processing is applied. If the secParamIndicator field bit DP is TRUE, the value of the disclosedKey field shall be the key from a previous time interval according to the key disclosure delay. The key disclosure delay shall be provided by the key management protocol and stored inside the SAD. The length of the field in octets (denoted “D” in Table 131) is related to the utilized

ICV calculation algorithm and shall be provided by the key management and stored in the SAD. Key disclosure should be performed in non-PTP messages. If DP is FALSE, the value of D shall be 0.

NOTE—An inclusion of this field in just specific messages will change the PTP message length, and thus, it might indirectly change delays in the message transport. This aspect needs to be taken into consideration when deciding in which PTP message the actual disclosed key is transmitted.

16.14.3.7 sequenceNo (UInteger S)

This field is optional. If present, it shall have a data type of UInteger with a length of S octets. For this edition, the value of SP in the secParamIndicator field shall be FALSE and the value of S shall be 0; that is, the length of the field is 0.

Note that the current approach relies on the sequenceId as part of the PTP header, which is also included in the integrity protection. The optional field is provided to enable later editions to enhance the sequenceId information to better support the detection of replay attacks.

16.14.3.8 RES (UInteger R)

This field is optional. If present, it shall have a data type of UInteger with a length of R octets. For this edition, the value of RP in the secParamIndicator field shall be FALSE and the value of RP shall be 0.

16.14.3.9 ICV (UInteger K)

ICV shall have a data type of UInteger with a length of K octets. The value of K depends on the algorithm as discussed below but must be an even integer.

This field shall contain the calculated ICV of the PTP message carrying the AUTHENTICATION TLV.

NOTE—There can be more than one AUTHENTICATION TLV attached to a PTP message.

The ICV is calculated according to the IntegrityAlgTyp provided by the key management. The selected IntegrityAlgTyp also determines the length of the ICV. The length shall also be provided by the key management.

Implementations should implement at least HMAC-SHA256-128, defined in FIPS PUB 198-1, for ICV calculation, as follows:

- HMAC-SHA256-128 specifies the use of SHA-256 defined in FIPS PUB 180-4, combined with HMAC defined in FIPS PUB 198-1. The output is truncated to 128 bits, as per IETF RFC 4868 (2017). The key size is the size of the hash value produced by SHA-256 (256 bits). The OID for HMAC-SHA-256 is 1.2.840.113549.2.9.

Other algorithms used for ICV calculation may be signaled using the associated algorithm OID values in the context of the key management.

16.14.4 PTP message processing

16.14.4.1 General requirements

Subclause 16.14.4 includes the various requirements for handling of PTP messages including an AUTHENTICATION TLV.

If multiple AUTHENTICATION TLVs are included in the PTP message, they shall be handled according to the parameters in the related SA.

Policy-limiting attributes are those attributes used to ascertain the specific security policy and SA applicable to a PTP message. These attributes are the sourcePortIdentity, domainNumber, majorSdoId and minorSdoId, and messageType fields of the PTP common header, as well as possibly the portIdentity of the port the message is sent to (i.e., the portIdentity attribute of the receiving port in the receiver).

The handling of the AUTHENTICATION TLV is determined by the negotiated security association. Therefore, in general, an interaction with the SPD and the SAD might be necessary on a per PTP message basis depending on the implementation.

For the processing of the AUTHENTICATION TLV, the sender and the receiver shall utilize the policy-limiting fields in querying the SPD regarding the security policy applicable for this PTP message. The query must result in an indication if this message requires secure processing or not. Note that a PTP instance might have one or multiple different security associations with different PTP Instances at the same time, depending on the applicable security policies.

If the message requires secure processing, the query must provide the list of SPPs applicable for this message (e.g., if message is to be secured by both immediate and delayed security processing, by appending two security TLVs, the SPPs for the two applicable security policies must be retrieved).

Each SPP shall be used to obtain from the SAD the relevant associated SA, and the implementation shall process the AUTHENTICATION TLV per the procedures described in the following subclauses. A Transparent Clock that does not have an SA for an incoming SPP may retransmit the AUTHENTICATION TLV unchanged, per 14.1. This allows Transparent Clocks incapable of processing the TLV to pass it unchanged.

PTP Instances not supporting the security TLV option shall ignore the AUTHENTICATION TLV. If such PTP Instances update nonmutable fields (and specifically the PTP message header), this might prohibit the receiver from successfully verifying a received ICV. This limitation must be taken into account during design and deployment of the AUTHENTICATION TLV option in the PTP network.

16.14.4.2 General Interactions with SAD and SPD

As the AUTHENTICATION TLV utilizes existing information from the PTP header, it also relies on the correct processing of these values if they are changed during the SA lifetime. This specifically applies to the sequenceId. According to 7.3.7, each port on a PTP Ordinary Clock or a Boundary Clock maintains a separate sequenceId pool for each message type sent to a destination address. Each port on a Transparent Clock must maintain a separate sequenceId pool for each type of Signaling, PTP management, and Pdelay_Req messages that originate at the Transparent Clock, according to 7.3.7.

To detect replay attacks, for each SA, the received sequenceId shall be monitored for each of the applicable message types which have separate sequenceId pools. The PTP instance shall discard an authenticated PTP message for which the sequenceId appears to not advance (with a roll-over event being considered an

advance of the sequenceId). For message types for which the sequenceId is associated with a received message (e.g., a FOLLOW_UP), only the first authenticated copy of a message with a given recent sequence ID shall be passed for further processing.

16.14.4.2.1 Inbound PTP messages

The implementation finds the security policy based on the policy-limiting fields of the incoming PTP messages. This policy specifies whether the PTP message requires AUTHENTICATION TLV processing and by what mechanism(s) it is secured. Once this is established, the implementation processes or drops the PTP message. If the PTP message is to be processed for security, then the implementation checks for the existence of the required AUTHENTICATION TLV(s) and uses the SPP in the TLV(s) to determine the SA parameters it should use to process the TLV(s). If not all the protection mechanisms, for example, AUTHENTICATION TLVs, required in the SPD exist in the message, the implementation shall discard the PTP message. If the appropriate security TLVs exist, the implementation processes the TLV(s) per PTP message processing descriptions later in this clause. The implementation also needs to verify that the received sequenceId is larger, after accounting for roll over, than the latest received sequenceId stored in the SAD. If this condition is true, the information about the latest received sequenceId is updated in the SAD.

16.14.4.2.2 Outbound PTP messages

The outbound security policy associated with the policy-limiting fields is retrieved from the SPD. This policy includes whether the PTP message must be unprotected, or protected, and by what authentication mechanism(s) (delayed or immediate). Once this is established, the implementation generates the required TLV(s) per the PTP message processing of 16.14.4.3.

When a delayed AUTHENTICATION TLV is used within the PTP Network and the network includes Transparent Clocks that retransmit messages, care must be taken to avoid the need for Transparent Clocks to update immutable fields. For example, the case where a Transparent Clock needs to change a one-step ingress message to a two-step egress message must be avoided.

16.14.4.3 Outbound PTP message processing

If an implementation generating an outgoing PTP message supports processing of the AUTHENTICATION TLV and the security policy requires that the PTP message be sent with a AUTHENTICATION TLV, the implementation shall append the AUTHENTICATION TLV as the last TLV in the message, unless a different location is specified in the applicable PTP profile.

16.14.4.3.1 Outgoing PTP message retransmission by Transparent Clocks

An implementation retransmitting PTP messages determines the applicable SA via lookup in the SAD based on the SPP in the received PTP message and processes the AUTHENTICATION TLV as indicated by the SA.

If implementations retransmitting a PTP messages support processing of the AUTHENTICATION TLV and the security policy requires that the PTP messages be sent with a AUTHENTICATION TLV associated with the retransmitter, the implementation updates the content of the AUTHENTICATION TLV attached to the ingress PTP message if immediate security processing is performed. If the AUTHENTICATION TLV in the ingress PTP message is associated with delayed security processing, the implementation does not change the delayed security TLV. A delayed AUTHENTICATION TLV shall be placed before an immediate AUTHENTICATION TLV if both are included in the PTP message.

NOTE—This will allow the retransmitting Transparent Clocks to update the ICV in the immediate AUTHENTICATION TLV without invalidating the ICV included within the delayed AUTHENTICATION TLV. TLVs that might be updated by Transparent Clocks need to be placed after a delayed AUTHENTICATION TLV since otherwise a retransmitting Transparent Clock will not be able to correctly update the included ICV.

The sequenceId shall be handled according to 7.3.7.

16.14.4.3.2 Integrity check value calculation

The ICV of an AUTHENTICATION TLV shall be computed over the PTP header, body, and TLVs preceding the AUTHENTICATION TLV and the AUTHENTICATION TLV itself up to the bytes including the ICV.

The ICV calculation shall not include the ICV of the AUTHENTICATION TLV itself. However, it shall include the correctionField if required by the security policy. If the security policy does not require the inclusion of the correctionField, the ICV calculation shall use the value 0 (zero) instead of the value of correctionField when computing the ICV.

If a retransmitter changes any field that is defined by the security policy to be immutable (e.g., correctionField), the retransmitter shall recompute the ICV and update the ICV value in the existing AUTHENTICATION TLV. Note that the recalculation of the ICV will effectively result in a new AUTHENTICATION TLV, which replaces the existing AUTHENTICATION TLV if immediate security processing is applied.

NOTE—The security policy defines values as immutable if they cannot be updated by a PTP instance securely. As an example, the correctionField value might not be updated by a Transparent Clock if the Transparent Clock does not possess the key to the corresponding SA. Changing an immutable field without the ability to recalculate the integrity check value invalidates the security of the message.

16.14.4.3.3 PTP message retransmission

To begin processing any PTP message (see Figure 50), the implementation identifies the SA associated with the SPP. Based on the security policy associated with the SPP, the implementation identifies the integrity algorithm type and key used to compute the ICV. If the PTP message is changed, then the implementation computes and updates the ICV. The update of the PTP message shall be done if the SA indicates immediate security processing for the associated SPP. If the PTP message is not changed, then the implementation continues PTP message processing.

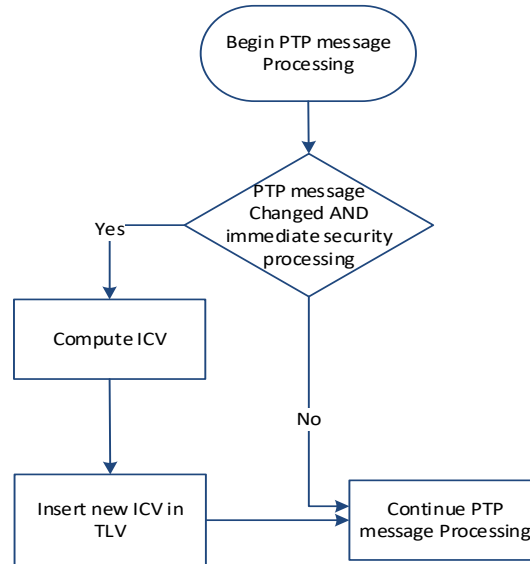


Figure 50—SecurityTLV retransmit immediate security processing

16.14.4.3.4 Event PTP message transmit case

To begin PTP message processing (see Figure 51), the implementation identifies the SA associated with the SPP. Based on the SA associated with the SPP, the implementation identifies the integrity algorithm type used to compute the ICV. Next, the implementation generates and appends an AUTHENTICATION TLV with the SPP value provided; generates the timestamp; updates any PTP message fields; generates the ICV; and inserts the ICV value into the ICV field of the AUTHENTICATION TLV.

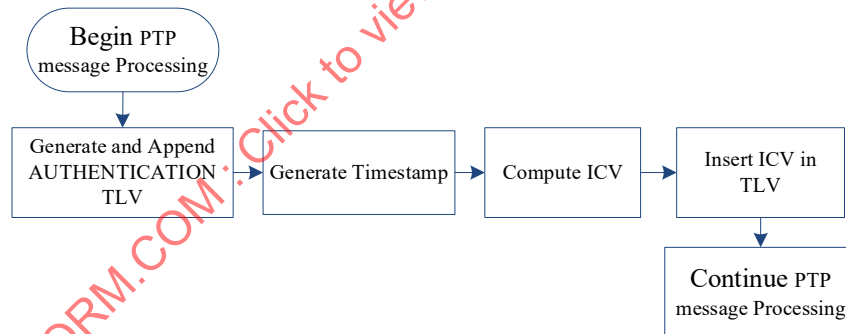


Figure 51—General AUTHENTICATION TLV transmit event processing

16.14.4.4 Inbound PTP message processing

If more than one AUTHENTICATION TLV is present in a PTP message, the initial AUTHENTICATION TLV applies only to the PTP message header and payload, including any preceding nonsecurity TLVs. The subsequent AUTHENTICATION TLV authenticates the PTP message header, payload, and any preceding TLVs including the previous AUTHENTICATION TLVs.

If only a single AUTHENTICATION TLV is present in a PTP message, it applies to the PTP message header and payload, including any preceding nonsecurity TLVs.

If the inbound message has an associated SPP in the SPD, the incoming message is expected to contain a AUTHENTICATION TLV with this SPP. An implementation shall enforce the corresponding SA and shall

drop all messages not secured with an associated AUTHENTICATION TLV or misauthenticated messages. The decision to drop PTP messages should be enforced on PTP message receipt and before any PTP message processing, other than header processing, takes place.

16.14.4.4.1 Integrity check value calculation

The ICV shall be computed over the PTP header and payload, including any preceding nonsecurity TLVs and security TLV, with the possible exception of the correctionField. The ICV calculation shall not include the ICV. If the security policy requires that the ICV include the correctionField, the ICV shall be computed over the entire PTP header and payload. If the security policy specifies that the correctionField is mutable, for example, in the case of delayed security processing, the value zero shall be used in the calculation of the ICV.

16.14.4.4.2 PTP message processing:

The following specifications shall apply when the SPP indicates that the message must be secured.

16.14.4.4.2.1 Immediate security processing case

To begin PTP message processing (see Figure 52), the implementation verifies that the AUTHENTICATION TLV is present. If the AUTHENTICATION TLV is not present, the implementation may send an alert and shall drop the PTP message. If the AUTHENTICATION TLV is present, the implementation shall verify that the SPP field in the message is known (i.e., there exists an associated entry in the SPD) and expected, based on the policy-limiting attributes. If the SPP is not known, the implementation may send an audit alert and shall drop the PTP message. The SA associated with the SPP is looked up in the SAD. The implementation computes an ICV using the security association parameters (e.g., integrity algorithm) as specified in the SA. If the computed ICV matches the received ICV, the implementation continues PTP message processing.

In the context of this security subclause, the word “drop” indicates that the content of the message must not affect the operation of the PTP protocol.

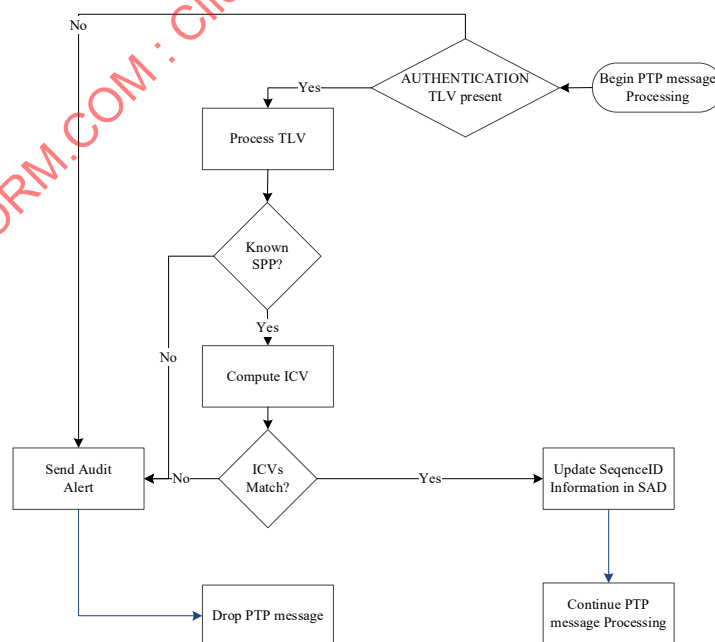


Figure 52— AUTHENTICATION TLV receive immediate security processing

16.14.4.4.2.2 Delayed security processing case

The PTP message processing with delayed security can be performed following different security policies which are:

- a) **Unauthenticated mode:** Store PTP messages for later verification, and continue normal processing of PTP messages. The unauthenticated mode should only be applied if:
 - 1) The application can roll back consequences of utilizing corrupted PTP information in case the delayed security check reveals that the PTP message was forged, and/or
 - 2) Additional security measures were taken (e.g., application of additional authentication mechanism).
- b) **Authenticated mode:** Store PTP messages until key is disclosed and perform PTP processing after successful verification of integrity.

To begin PTP message processing (see Figure 53), the implementation verifies that the AUTHENTICATION TLV is present. If the AUTHENTICATION TLV is not present, the implementation may send an alert and shall drop the PTP message. If the AUTHENTICATION TLV is present, the implementation verifies the SPP is known. If the SPP is not known, the implementation may send an alert and shall drop the PTP message. If the SPP is known, the implementation performs the security checks associated with the particular delayed security scheme. If the PTP message does not pass the security checks, the implementation may send an alert and shall drop the PTP message. If the PTP message passes the security checks, the implementation stores the PTP message for later authentication.

The implementation verifies whether or not the AUTHENTICATION TLV contains a disclosed key:

- If the AUTHENTICATION TLV does not contain a disclosed key, the implementation verifies the policy mode of the PTP message. In case the policy mode is unauthenticated, the implementation continues with PTP message processing. In case the policy mode is authenticated, the implementation shall wait with the PTP message processing until the authenticity of the PTP message (ICV test) is verified.
- If the AUTHENTICATION TLV contains a disclosed key, the implementation computes the ICV over each stored PTP message that is associated with the disclosed key.
 - If for a stored PTP message, the computed ICV matches the received ICV, the implementation verifies that the policy mode is authenticated. If the policy mode is authenticated, the implementation continues PTP message processing. If the policy mode is unauthenticated, the implementation may optionally log the authenticity of the PTP message.
 - If the computed and received ICV for a stored PTP message do not match, then:
 - If the mode is authenticated, the implementation may send an alert and shall drop the stored PTP message.
 - If the mode is unauthenticated, the implementation may send an alert and should perform application-specific roll-back actions to attempt to recover from the possible attack.

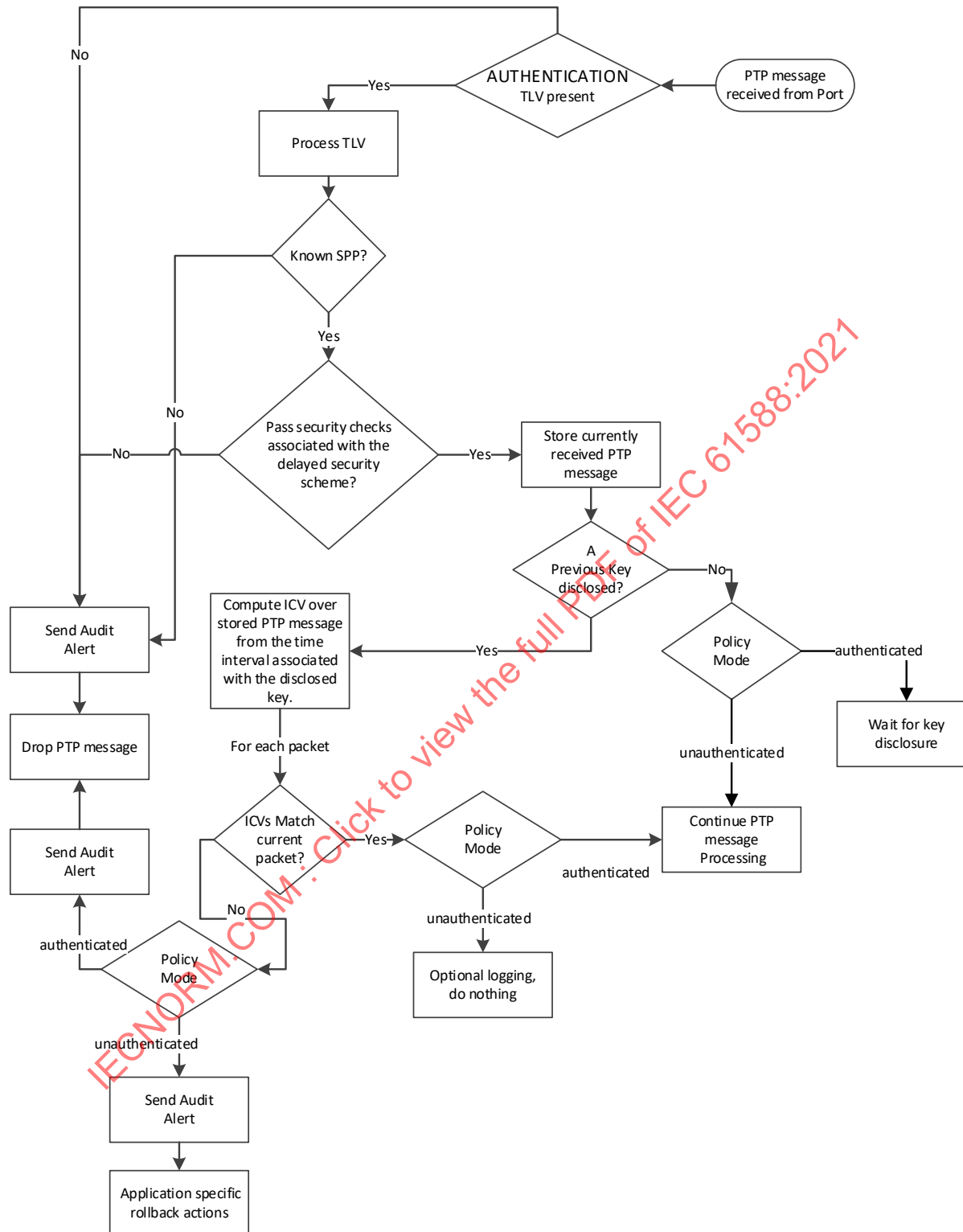


Figure 53—AUTHENTICATION TLV receive delayed security processing

17. State configuration options

17.1 General

This clause specifies additional optional features that may be used to enhance performance of the best master clock algorithm and/or to exert more control over the selection of PTP Port state.

Many applications require one or more of the following capabilities:

- Automatic recovery from a break in the communication network
- Automatic recovery from the failure of a PTP Instance
- Explicit control over the selection of PTP Port state

This standard provides several features designed to meet these requirements.

The operation of the best master clock algorithm and state machine in Ordinary Clocks and Boundary Clocks (Clause 9) ensures that a master–slave hierarchy is established, with the best PTP Instance present in the PTP Network being the Grandmaster PTP Instance. This provides automatic recovery from both PTP Network failure and failure of individual PTP Instances. The rate of recovery is dependent on the announceInterval and on the topology of the PTP Network.

Peer-to-peer Transparent Clocks (see Clause 10) provide for rapid recovery in the event of network reconfiguration. Since the peer-to-peer delay mechanism (see 11.4) measures the path delays on all PTP Links in the event of a reconfiguration, the needed path delay correction information is immediately available.

NOTE—It is recommended that the options specified in this clause be used with care. For example, if some PTP Instances connected to a PTP Communication Path are configured to use the Acceptable Master Table and some are not, it is possible that more than one PTP Port will consider itself to be the best master. If PTP Instances connected to a PTP Communication Path are configured with incompatible Acceptable Master Tables, it is possible that more than one PTP Port will consider itself to be the best master. Similar misconfiguration inconsistencies can occur with the configuration mechanism of any of these options.

17.2 Grandmaster clusters (optional)

17.2.1 General specification

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile or by the manufacturer, then the option shall operate as specified in this subclause.

This option specifies a mechanism for faster selection of the Grandmaster PTP Instance from the set of PTP Instances for which this option is both implemented and enabled. The option is enabled on a PTP Instance by configuration of the grandmasterClusterDS.actualTableSize member as specified in 17.2.3.3.2 and 17.2.2.

The grandmaster cluster option requires unicast transmissions between members of the grandmaster cluster. When the normal operation of the best master clock algorithm is used, the time to recover from the failure of a Grandmaster PTP Instance depends on the announceInterval. If the required time to change from one Grandmaster Clock to another is incompatible with the multicast announceInterval, the mechanism of 17.2 may be used to decrease the time required to select a new Grandmaster PTP Instance. If this option is implemented, the unicast negotiation option (see 16.1) shall also be implemented.

Two to six Ordinary Clocks or Boundary Clocks may be designated as a grandmaster cluster. For correct operation, each PTP Instance in the cluster should be configured with values of `defaultDS.priority1` such that the `defaultDS.priority1` values of all members of the cluster are less than the `defaultDS.priority1` values of all other PTP Instances in the domain.

NOTE—Although designed for use in improving the changeover time of Grandmaster PTP Instances, in some topologies, this mechanism can be useful for designating clusters of potential Grandmaster PTP Instances below the Grandmaster PTP Instance in the master–slave hierarchy. The configuring of clusters of potential Grandmaster PTP Instances for such use is outside of the scope of this standard.

17.2.2 Operation of the grandmaster cluster

Each PTP Instance in the cluster shall:

- a) Maintain a configured array containing the port addresses of potential Grandmaster PTP Instances, in the `grandmasterClusterDS` data set (see 17.2.3). The members of the array shall each hold the port address of one PTP Port of a member of the grandmaster cluster to enable unicast transmission between members.
- b) Use the unicast message negotiation option (see 16.1) to periodically request unicast Announce messages from all the PTP Ports listed in the `grandmasterClusterDS.portAddress` array.
- c) If a PTP Port is transmitting a unicast Announce message under the terms of this subclause and is in the MASTER state, set the `flagField.alternateMasterFlag` to FALSE; otherwise set the `flagField.alternateMasterFlag` to TRUE.
- d) Insert the value of `grandmasterClusterDS.logQueryInterval` member into the `logInterMessageInterval` field of the REQUEST_UNICAST_TRANSMISSION TLV.
- e) Request a renewal prior to the expiration of each negotiated unicast transmission.
- f) Use the received unicast Announce messages from the cluster members, irrespective of the value of `alternateMasterFlag`, in addition to the normal Announce messages describing the Grandmaster Clock to exercise the best master clock algorithm to determine the `portState` of each of its PTP Ports.

If the member `grandmasterClusterDS.portAddress` is empty (`actualTableSize` member is 0), this option shall be inactive except for the management of `grandmasterClusterDS`.

17.2.3 grandmasterClusterDS data set specifications

17.2.3.1 General

The members of this data set are as follows:

- `grandmasterClusterDS.maxTableSize`
- `grandmasterClusterDS.logQueryInterval`
- `grandmasterClusterDS.actualTableSize`
- `grandmasterClusterDS.portAddress`

This data set is contained within the data sets of the PTP Instance (see 8.2.11).

17.2.3.2 Static members of the grandmasterClusterDS data set**17.2.3.2.1 grandmasterClusterDS.maxTableSize**

This member shall be the maximum permitted value of the actualTableSize member. The data type is UInteger8. The value shall be 5.

17.2.3.3 Configurable members of the grandmasterClusterDS data set**17.2.3.3.1 grandmasterClusterDS.logQueryInterval**

The value of logQueryInterval shall be the logarithm to the base 2 of the mean interval in seconds between unicast Announce messages from cluster members. The data type is Integer8. The default value is specified by the applicable PTP Profile.

17.2.3.3.2 grandmasterClusterDS.actualTableSize

This member shall specify the number of port addresses contained in the array of the portAddress member. The data type is UInteger8.

The default value of actualTableSize shall be 0 unless otherwise specified in the applicable PTP Profile.

Management write of value 0 to actualTableSize shall cause the PTP Instance to clear all previous portAddress members from the array.

Management write of a value to actualTableSize that is greater than maxTableSize shall return a management error.

17.2.3.3.3 grandmasterClusterDS.portAddress

This member shall specify the array of port addresses of each member of the grandmaster cluster. The data type shall be PortAddress[actualTableSize] (see 5.3.6).

17.3 Alternate master (optional)**17.3.1 General**

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile or by the manufacturer, then the option shall operate as specified in this subclause.

This option specifies a mechanism for PTP Ports on a PTP Communication Path that are not currently the MASTER port of that PTP Communication Path, that is, alternate Master PTP Ports, to exchange PTP timing information with other PTP Ports on the same PTP Communication Path, and for each of the other PTP Ports to acquire knowledge of the characteristics of the transmission path between itself and each alternate Master PTP Port. This option is disabled by setting the value of alternateMasterPortDS.numberOfAlternateMasters to 0 (see 17.3.3.2.1). The feature allows a PTP port to monitor an alternate Master PTP Port.

17.3.2 Transmission of messages by alternate masters

A PTP Port shall transmit Announce messages subject to the restrictions defined by the values of the members of the `alternateMasterPortDS` (see 17.3.3). A PTP Port transmitting Announce messages in accordance with the restrictions of 17.3.3 shall set the `alternateMasterFlag` (see 13.3.2.8) to TRUE. These messages shall be transmitted at the interval defined by `portDS.logAnnounceInterval` (see 8.2.15.4.1).

A PTP Port shall transmit Sync and, if a two-step PTP Port, Follow_Up messages subject to the restrictions defined by the values of the members of the `alternateMasterPortDS` (see 17.3.3). A PTP Port transmitting Sync and Follow_Up messages in accordance with the restrictions of 17.3.3 shall set the `alternateMasterFlag` to TRUE. These messages shall be transmitted at the interval defined by the value of `alternateMasterPortDS.logAlternateMulticastSyncInterval` (see 17.3.3.2.3).

A PTP Port transmitting Delay_Resp messages under the terms of 17.3 shall set the `alternateMasterFlag` to TRUE.

NOTE—A Slave PTP Instance that does not want to use information from alternate Master PTP Instances merely ignores all messages with `alternateMasterFlag` TRUE.

17.3.3 `alternateMasterPortDS` data set specifications

17.3.3.1 General

The members of this data set are as follows:

- `alternateMasterPortDS.numberOfAlternateMasters`
- `alternateMasterPortDS.transmitAlternateMulticastSync`
- `alternateMasterPortDS.logAlternateMulticastSyncInterval`

This data set shall be supported by a PTP Port that supports the alternate master mechanism.

17.3.3.2 Configurable members of the `alternateMasterPortDS` data set

17.3.3.2.1 `alternateMasterPortDS.numberOfAlternateMasters`

The `alternateMasterPortDS.numberOfAlternateMasters` limits the number of PTP Ports that can simultaneously transmit messages with the alternate master flag set to TRUE.

A PTP Port, PTP Port-A not in the MASTER state, shall transmit multicast Announce messages when the number of other PTP Ports that:

- Are currently transmitting qualified (see 9.3.2.5) Announce messages, with `flagField.alternateMasterFlag` TRUE, that are being received by PTP Port-A, and
- Would be chosen using the best master algorithm as best master in preference to PTP Port-A,
- Is less than the value of `alternateMasterPortDS.numberOfAlternateMasters`.

The data type shall be `UInteger8`. The default value for `alternateMasterPortDS.numberOfAlternateMasters` shall be 0.

NOTE—The default value of `alternateMasterPortDS.numberOfAlternateMasters` causes multicast Announce messages to be transmitted only when the PTP Port is in the MASTER state.

17.3.3.2.2 **alternateMasterPortDS.transmitAlternateMulticastSync**

The `alternateMasterPortDS.transmitAlternateMulticastSync` controls Sync transmission. If TRUE and the PTP Port is currently transmitting multicast Announce messages with `alternateMasterFlag` TRUE, the PTP Port shall also transmit multicast Sync and, if a two-step PTP Instance, Follow_Up messages. Otherwise do not transmit these messages.

The data type shall be Boolean. The default value is specified by the applicable PTP Profile.

17.3.3.2.3 **alternateMasterPortDS.logAlternateMulticastSyncInterval**

The value of `alternateMasterPortDS.logAlternateMulticastSyncInterval` shall be the logarithm to the base 2 of the mean interval in seconds between Sync messages transmitted under the terms of 17.3.

The data type shall be Integer8. The default value is specified by the applicable PTP Profile.

17.3.4 **Transmission of Delay_Req**

A PTP Port may transmit Delay_Req messages under the terms of 17.3. These messages have `alternateMasterFlag` set to FALSE (see 13.3.2.8). These Delay_Req messages may be transmitted even if the PTP Port state is not SLAVE. These Delay_Req messages may be transmitted even if the PTP Port is not configured to become an Alternate Master. These Delay_Req messages may be transmitted even if no Sync messages have been received or are not currently being received on the PTP Port. These Delay_Req messages shall be transmitted at the interval defined by `portDS.logMinDelayReqInterval`.

The choice to send Delay_Req messages under the terms of 17.3 is implementation specific.

17.3.5 **Transmission of Delay_Resp**

A PTP Port may transmit Delay_Resp messages under the terms of 17.3 in response to received Delay_Req messages.

17.3.6 **Summary of Messages for Alternate Master Option**

Under the terms of this option, a PTP Port can be both a “sender” and a “receiver.”

A sender is a PTP Port that sends PTP Instance information to another PTP Port. A receiver is a PTP Port that receives PTP Instance information from another PTP Port.

Table 133 summarizes the alternate master semantics for senders.

Table 133—Message examples for sender

Message	Direction	alternateMasterFlag	Additional Note
Announce	Egress	TRUE	Subject to restrictions defined by the values of the members of the alternateMasterPortDS (see 17.3.3).
Sync	Egress	TRUE	Subject to restrictions defined by the values of the members of the alternateMasterPortDS (see 17.3.3).
Delay_Req	Ingress	FALSE expected	Might generate a Delay_Resp.
Delay_Resp	Egress	TRUE	

Table 134 summarizes the alternate master semantics for receivers.

Table 134—Message examples for receiver

Message	Direction	alternateMasterFlag	Additional Note
Announce	Ingress	TRUE expected	Can be ignored by receiver if monitoring is not desired. Not used in the BMCA.
Sync	Ingress	TRUE expected	Can be ignored by receiver if monitoring is not desired.
Delay_Req	Egress	FALSE	Can be sent even if not in the SLAVE state.
Delay_Resp	Ingress	TRUE expected	Can be ignored by receiver if monitoring is not desired.

17.4 Unicast discovery (optional)

17.4.1 General

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile or by the manufacturer, then the option shall operate as specified in this subclause.

This option specifies a mechanism for PTP to be used over a network that does not provide multicast (for example, many IP networks). This option is disabled by setting the value of unicastDiscoveryPortDS.maxTableSize to 0 (see 17.4.3.2.1). A PTP Instance is configured with the addresses of PTP Ports of other PTP Instances with which it should attempt to establish unicast communication. The PTP Instance may request that these PTP Ports transmit unicast Announce, Sync, and Delay_Resp messages to it. If this option is implemented, the unicast negotiation option (see 16.1) shall also be implemented.

17.4.2 Operation of unicast discovery

An Ordinary Clock or Boundary Clock shall maintain a configured table of PTP Ports of potential Master PTP Instances, specified by the unicastDiscoveryPortDS data set of 17.4.3. The portAddress members of the table each hold the protocol address of a remote PTP Port with which this PTP Instance may attempt to establish communication.

The PTP Instance shall use the unicast message negotiation option (see 16.1) to periodically request unicast Announce messages from all the PTP Ports listed in the portAddress members. If a request is not granted by a PTP Port, the request shall be repeated after the delay indicated by the logQueryInterval member of the table.

If the table is empty (actualTableSize member is 0), this option shall have no effect except for management of unicastDiscoveryPortDS.

17.4.3 unicastDiscoveryPortDS data set specifications

17.4.3.1 General

The members of this data set are as follows:

- unicastDiscoveryPortDS.maxTableSize
- unicastDiscoveryPortDS.logQueryInterval
- unicastDiscoveryPortDS.actualTableSize
- unicastDiscoveryPortDS.portAddress

17.4.3.2 Static members of the unicastDiscoveryPortDS data set

17.4.3.2.1 unicastDiscoveryPortDS.maxTableSize

This member shall return the maximum permitted value of the actualTableSize member. The data type shall be UInteger16. The default value is implementation specific, determined by the product.

17.4.3.3 Configurable members of the unicastDiscoveryPortDS data set

17.4.3.3.1 unicastDiscoveryPortDS.logQueryInterval

The value of logQueryInterval shall be the logarithm to the base 2 of the mean interval in seconds between requests from a PTP Instance for a unicast Announce message. The data type shall be Integer8. The default value is specified by the PTP Profile.

17.4.3.3.2 unicastDiscoveryPortDS.actualTableSize

This member shall specify the number of protocol addresses contained in the portAddress member. The data type shall be UInteger16.

The default value of actualTableSize is 0 unless otherwise specified in the applicable PTP Profile.

Management write of value 0 to actualTableSize shall cause the PTP Port to clear all previous portAddress members from the array.

Management write of a value to actualTableSize that is greater than maxTableSize shall return a management error.

17.4.3.3.3 unicastDiscoveryPortDS.portAddress

This member shall specify the array of protocol addresses, of size actualTableSize. Each protocol address in the array shall use the PortAddress data type (see 5.3.6).

17.5 Acceptable master table (optional)

17.5.1 General

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile or by the manufacturer, then the option shall operate as specified in this subclause.

This option specifies a mechanism that allows PTP Ports in the SLAVE state to be configured to refuse to synchronize to PTP Instances not on the acceptable master list. This option is disabled by setting the value of `acceptableMasterTableDS.maxTableSize` to 0 (see 17.5.3.3.1).

NOTE—This can be used to rule out synchronization to suspected rogue or spurious Master PTP Instances.

17.5.2 Operation of the acceptable master table

An Ordinary Clock or Boundary Clock shall maintain a configurable data set, `acceptableMasterTableDS`, and the per PTP Port data set `acceptableMasterPortDS`, which contains the configurable Boolean value `acceptableMasterPortDS.enable`.

The `acceptableMasterTableDS.list` values shall carry the respective `acceptablePortIdentity` and `alternatePriority1` values of each member of acceptable master table. The value `FFFF16` used in the `portNumber` portion of `acceptablePortIdentity` (see 17.5.3.2) shall indicate that all PTP Ports on the PTP Instance specified by the `clockIdentity` portion of `acceptablePortIdentity` are acceptable.

17.5.3 `acceptableMasterTableDS` data set specifications

17.5.3.1 General

The members of this data set are as follows:

- `acceptableMasterTableDS.maxTableSize`
- `acceptableMasterTableDS.actualTableSize`
- `acceptableMasterTableDS.list`

This data set is contained within the data sets of the PTP Instance (see 8.2.12).

17.5.3.2 AcceptableMaster derived data type

The `AcceptableMaster` data type represents each element in the data set's list, `acceptableMasterTableDS.list` (see 17.5.3.4.2).

```
Struct AcceptableMaster
{
    PortIdentity acceptablePortIdentity;
    UInteger8 alternatePriority1;
};
```

NOTE—The 2008 edition of this standard used a different `AcceptableMaster` data type for PTP management messages, and that data type is deprecated (see 15.5.3.3.14.2). The deprecated `AcceptableMaster` data type does not apply to non-PTP management mechanisms.

17.5.3.3 Static members of the acceptableMasterTableDS data set

17.5.3.3.1 acceptableMasterTableDS.maxTableSize

This member shall return the maximum permitted value of the actualTableSize member. The data type shall be UInteger16. The default value is implementation specific, determined by the product.

17.5.3.4 Configurable members of the acceptableMasterTableDS data set

17.5.3.4.1 acceptableMasterTableDS.actualTableSize

This member shall specify the number of AcceptableMaster elements contained in the list member. The data type shall be UInteger16.

The default value of acceptableMasterTableDS.actualTableSize shall be 0 unless otherwise specified in the applicable PTP Profile.

Management write of value 0 to acceptableMasterTableDS.actualTableSize shall clear all previous elements from the list member.

Management write of a value to acceptableMasterTableDS.actualTableSize that is greater than maxTableSize shall return a management error.

17.5.3.4.2 acceptableMasterTableDS.list

The list member provides the list of acceptable masters in the PTP Instance (see 17.5.2). Each element of the list shall use the data type AcceptableMaster.

Elements in the list can be created or deleted if those operations are supported by the management mechanism.

Individual items within each AcceptableMaster data type (e.g., alternatePriority1) can be read or written if those operations are supported by the management mechanism.

If management write is supported for AcceptableMaster items acceptablePortIdentity and alternatePriority1, the value for both items shall be provided within a single write operation, and the update of both items shall be atomic. If any of the two values fails to update, a management error shall be returned.

17.5.4 acceptableMasterPortDS data set specifications

17.5.4.1 General

The members of this data set are as follows:

acceptableMasterPortDS.enable

This data set is contained within the data sets of the PTP Port (see 8.2.22).

17.5.4.2 Configurable members of the acceptableMasterPortDS data set

17.5.4.2.1 acceptableMasterPortDS.enable

The operation of the acceptable master table option on each PTP Port shall be as specified in Table 135.

Table 135—Operation of acceptable master table option

acceptableMasterPortDS.enable	Operational specification
FALSE	The acceptable master table option is not used on this PTP Port. The normal operation of the protocol is in effect. The PTP Port shall continue to process management of acceptable master data sets (acceptableMasterTableDS and acceptableMasterPortDS).
TRUE	The PTP Port indicated by the value of $E_{r\text{best}}$ determined by the best master clock algorithm (see 9.3) shall be a member of the acceptable master table (i.e., acceptableMasterTable.list). If qualified Announce messages (see 9.3.2.5) are being received from more than one member of the acceptable master table, the data set comparison algorithm of 9.3.4 shall be used to select $E_{r\text{best}}$ to determine the PTP Port selected as the master from the members of this table. If the alternatePriority1 member of the AcceptableMaster member of the table for a PTP Port is 0, the alternatePriority1 member shall have no effect on the computation of $E_{r\text{best}}$. If the value of the alternatePriority1 member is greater than 0, the value of grandmasterPriority1 in the Announce message from the remote PTP Port shall be replaced by the value of the alternatePriority1 member of this table for the purposes of computing $E_{r\text{best}}$.

The data type shall be Boolean. The default value of acceptableMasterPortDS.enable shall be FALSE unless otherwise specified in the applicable PTP Profile.

17.6 Mechanism for external configuration of a PTP Instance's PTP Port state (optional)

17.6.1 General specifications

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile or by the manufacturer, then the option shall operate as specified in this subclause.

This option specifies a mechanism for configuring the PTP state of a PTP Port via a mechanism specified outside the scope of this standard, with the value of the attribute defaultDS.externalPortConfigurationEnabled determining whether the external configuration option is used or not, and the value of the externalPortConfigurationPortDS.desiredState determining the desired PTP state of the respective PTP Port.

Informative examples of using this option are provided below:

- a) This option is supported “as-is” (e.g., in a PTP Instance implementing the High Accuracy Profile of L5):
 - 1) The value of portDS.portState is always equal to the value of the externalPortConfiguration PortDS.desiredState

- 2) The value of the externalPortConfigurationPortDS.desiredState is set by a mechanism specified outside of this standard, such as:
 - i) Implementation specific, for example, an external process
 - ii) Standard management mechanism, for example, SNMP [B27] and YANG [B22]
- b) This option is extended by applicable PTP Profiles or other documents outside this standard (see 17.6.5.2):
- 3) The extension to this option specifies when the value of portDS.portState is equal to the value of the externalPortConfigurationPortDS.desiredState and specifies the following exceptions:
 - i) Fault management:
 - i) Specifies conditions under which the PTP Instance detects a fault (see 7.6.2.1), which causes the portDS.portState to be set to FAULTY.
 - ii) Specify conditions under which the PTP Instance clears a fault. For example, specifying that clearing a fault results in restoring portDS.portState to the value of the externalPortConfigurationPortDS.desiredState.
 - ii) Limitation of the allowed PTP Port states:
 - i) Specifies which PTP Port states are not allowed and the results of attempting to set the value of externalPortConfigurationPortDS.desiredState to one of these disallowed values.
 - iii) Configuration by an internal mechanism:
 - i) Specify an internal mechanism that sets the value of the externalPort ConfigurationPortDS.desiredState. This mechanism could simply instantiate a predetermined set of PTP States, or it could be a dynamic mechanism based on internal conditions.
 - ii) Specify the response to a management write to the externalPortConfiguration PortDS.desiredState by the implemented mechanism.

17.6.2 defaultDS.externalPortConfigurationEnabled

When the value of the defaultDS.externalPortConfigurationEnabled is TRUE, this option is in an enabled state, as specified in 17.6.5. When the value of the defaultDS.externalPortConfigurationEnabled is FALSE, this option is in a disabled state, as specified in 17.6.4.

The data type of the defaultDS.externalPortConfigurationEnabled shall be Boolean. The classification is configurable.

The specification initialization value (see 8.1.3.4) shall be FALSE.

NOTE—After any initializing event, for example, power-up, if the initialization value of defaultDS.externalPortConfigurationEnabled is FALSE based on 8.1.3.4, the PTP Instance operates with PTP Port states being determined by the normal operation of the protocol's BMCA.

17.6.3 externalPortConfigurationPortDS

17.6.3.1 General

The externalPortConfigurationPortDS data set shall be implemented for each PTP Port (i.e., under portList of 8.1.4.2).

17.6.3.2 externalPortConfigurationPortDS.desiredState

When the value of the defaultDS.externalPortConfigurationEnabled is TRUE, the externalPortConfigurationPortDS.desiredState is used to transition the PTP Port's state (portDS.portState) to a desired value.

The data type shall be Enumeration8, using the values from Table 20 (portDS.portState values). The classification is configurable. The default value shall be PASSIVE unless otherwise specified.

17.6.4 Specifications for the disabled state of this option

While the value of the defaultDS.externalPortConfigurationEnabled is FALSE, this option is in the disabled state and the specifications of 17.6.5 shall not be in effect.

When the value of the attribute defaultDS.externalPortConfigurationEnabled transitions from TRUE to FALSE:

- The PTP Port states and data sets shall remain unchanged until modified by normal operation of the BMCA according to Clause 9.
- The announce receipt timeout mechanism of 9.2.6.12 shall be restarted on all PTP Ports except those in the INITIALIZING, PRE_MASTER, FAULTY, DISABLED, or MASTER states.

While the value of the attribute defaultDS.externalPortConfigurationEnabled is FALSE:

- All Clause 9 specifications and all other clauses of this standard, with the exception of 17.6.5, shall execute as stated. For example, the updating of PTP Port state and PTP Instance data sets of the PTP Instance is determined by the BMCA.

17.6.5 Specifications for the enabled state of this option

17.6.5.1 General

When the value of the defaultDS.externalPortConfigurationEnabled is TRUE, this option is in the enabled state and the specification of 17.6.5 shall be in effect.

17.6.5.2 Value of the portDS.portState when this option is enabled

When the value of the defaultDS.externalPortConfigurationEnabled is TRUE, the value of the member portDS.portState shall be set to the value of the member externalPortConfigurationPortDS.desiredState unless otherwise specified in the applicable PTP Profile or another document outside this standard. Such a PTP Profile or document may do the following:

- Define fault conditions under which portDS.portState is set to the FAULTY state regardless of the value of the externalPortConfigurationPortDS.desiredState
- Limit the allowed values for externalPortConfigurationPortDS.desiredState and specify the results of an attempt to set to a disallowed value
- Define how and when the value of portDS.portState transitions to the value of externalPortConfigurationPortDS.desiredState

17.6.5.3 Departures from normal operation of the protocol when this option is enabled

When the value of the `defaultDS.externalPortConfigurationEnabled` is `TRUE`, the following departures from normal operation of the protocol must be in effect:

- The state machines of Figure 30 or Figure 31 shall not be used.
- The Announce receipt timeout mechanism (see 9.2.6.12) shall not be active.
- Subclause 9.2.1 through subclause 9.2.3 shall not be in effect. If implemented, `defaultDS.slaveOnly` should be `FALSE`, and `portDS.masterOnly` should be `FALSE` on all PTP Ports of the PTP Instance.

NOTE 1—`defaultDS.slaveOnly` and `portDS.masterOnly` are not used when external port configuration is enabled.

- Subclause 9.2.4 shall continue to be in effect.
- The specifications of 9.2.5 (normal PTP Port state machine), 9.2.6 (events pertaining to the normal state machine), and 9.2.7 (applicability of normal PTP events) shall not be in effect.

NOTE 2—This and the previous point covers all of 9.2.

- The specifications of 9.3 shall not be in effect.

NOTE 3—The disabling of 9.3 disables all aspects of the BMCA.

- The specifications of 9.4 shall continue to be in effect.
- Subclause 9.5 shall continue to be in effect with the exception of the specifications of 9.5.2.3 and 9.5.3.
- Subclause 9.5.2.3 specifications shall not be permitted to be in effect.

NOTE 4—Subclause 9.5.2.3 discusses the receipt of a PTP message sent from the same PTP Instance. This situation is one of the possible misconfigurations when using this option.

- The specifications of 9.5.3 shall be replaced by the specifications of 17.6.5.5.
- The specifications of 9.6 shall continue to be in effect.
- The following options shall not be enabled: 17.2 Grandmaster clusters (this option uses the normal BMCA which is disabled); 17.5 Acceptable master table (this option uses the normal BMCA which is disabled); and 17.7 Reduced state sets and use of the `<foreignMasterList>` feature (this feature requires the use of Figure 30 or Figure 31).
- The specifications of all other subclauses shall remain in effect.

17.6.5.4 Updating data sets when this option is enabled

When the value of the `defaultDS.externalPortConfigurationEnabled` is `TRUE`, the relevant data sets of the PTP Instance are updated as follows:

- a) After a change of `portDS.portState` of any PTP Port, if none of the PTP Instance's PTP Ports are in the `SLAVE` or `UNCALIBRATED` state and at least one PTP Port is in the `MASTER`, `PRE_MASTER`, or `PASSIVE` state, that is, the PTP Instance is the Grandmaster PTP Instance, the PTP Instance's data sets shall be updated as specified in Table 136. In addition, while the PTP Instance is the Grandmaster PTP Instance, these data sets shall be updated as specified in Table 136, per 9.4 and 9.6, in response to any changes that affect the following:
 - 1) The `defaultDS`, for example, change in `clockQuality` of the PTP Instance, or
 - 2) The `timePropertiesDS`, for example, loss of contact with GNSS or notification of a pending leap second from GNSS.
- b) If a PTP Port of the PTP Instance is in the `SLAVE` or `UNCALIBRATED` state, that is, the PTP

Instance is not the Grandmaster PTP Instance, the PTP Instance's data sets shall be updated as specified in Table 137, based on the Announce message (A) most recently received by this PTP Port.

NOTE—A transient situation or an error by the entity using the external configuration mechanism can result in misconfiguration such as follows:

- None of the PTP Ports that share a PTP Communication Path are in the MASTER state, with the result that a PTP Port in the SLAVE or UNCALIBRATED state on that PTP Communication Path does not receive Announce messages.
- Multiple PTP Ports on a PTP Instance are in the SLAVE state, with a result that the PTP Instance's data sets are updated by multiple PTP Ports, possibly with different values, and thus, thrashing is observed.

Preventing such situations is outside the scope of this standard.

Table 136—Update of data sets when the PTP Instance is the Grandmaster PTP Instance as determined by the action of 17.6.5.2

Update this field	From the indicated source
currentDS data set	
currentDS.stepsRemoved	set to 0
currentDS.offsetFromMaster	set to 0
currentDS.meanDelay	set to 0
currentDS.synchronizationUncertain	Follow rules in 8.2.2.5
parentDS data set	
parentDS.parentPortIdentity	parentDS.parentPortIdentity.clockIdentity member set to the value of defaultDS.clockIdentity field. parentDS.parentPortIdentity.portNumber member is 0
parentDS.grandmasterIdentity	defaultDS.clockIdentity
parentDS.grandmasterClockQuality	defaultDS.clockQuality
parentDS.grandmasterPriority1	defaultDS.priority1
parentDS.grandmasterPriority2	defaultDS.priority2
parentDS.synchronizationUncertain	Follow rules in 8.2.3.11
timePropertiesDS data set	
timePropertiesDS.currentUtcOffset	Follow rules in 9.4
timePropertiesDS.currentUtcOffsetValid	Follow rules in 9.4
timePropertiesDS.leap59	Follow rules in 9.4
timePropertiesDS.leap61	Follow rules in 9.4
timePropertiesDS.timeTraceable	Follow rules in 9.4
timePropertiesDS.frequencyTraceable	Follow rules in 9.4
timePropertiesDS.ptpTimescale	Follow rules in 9.4
timePropertiesDS.timeSource	Follow rules in 9.4

Table 137—Update of data sets for a PTP Port in the SLAVE or UNCALIBRATED state as determined by the action of 17.6.5.2

Update this field	From the indicated source
currentDS data set	
currentDS.stepsRemoved	1 + value of stepsRemoved of A (A is defined in last row of table)
currentDS.synchronizationUncertain	Follow rules in 8.2.2.5
parentDS data set	
parentDS.parentPortIdentity	sourcePortIdentity of A
parentDS.grandmasterIdentity	grandmasterIdentity of A
parentDS.grandmasterClockQuality	grandmasterClockQuality of A
parentDS.grandmasterPriority1	grandmasterPriority1 of A
parentDS.grandmasterPriority2	grandmasterPriority2 of A
parentDS.synchronizationUncertain	Follow rules in 8.2.3.11
timePropertiesDS data set	
timePropertiesDS.currentUtcOffset	currentUtcOffset field of A
timePropertiesDS.currentUtcOffsetValid	The logical value of the currentUtcOffsetValid bit of the flagField of A
timePropertiesDS.leap59	The logical value of the leap59 bit of the flagField of A
timePropertiesDS.leap61	The logical value of the leap61 bit of the flagField of A
timePropertiesDS.timeTraceable	The logical value of the timeTraceable bit of the flagField of A
timePropertiesDS.frequencyTraceable	The logical value of the frequencyTraceable bit of the flagField of A
timePropertiesDS.ptpTimescale	The logical value of the ptpTimescale bit of the flagField of A
timePropertiesDS.timeSource	The value of the timeSource field of A
The source of the data for the update shall be the specified fields in the most recent Announce message (A) received on the PTP Port in the SLAVE or UNCALIBRATED state.	

17.6.5.5 Receipt of an Announce message from another PTP Instance

When the value of the defaultDS.externalPortConfigurationEnabled is TRUE, the receipt of an Announce message shall be as specified in this subclause and the specifications of 9.5.3 are not in effect, per 17.6.5.3.

The logic for processing an Announce message shall be as defined in Figure 54. The states indicated in this figure refer to the current state of the PTP Port receiving the Announce message.

If the PTP Port receiving an Announce message is in the INITIALIZING or DISABLED states, the PTP message shall be discarded. If the PTP Port receiving an Announce message is in the FAULTY state, the PTP message shall be discarded except for implementation-specific purposes.

NOTE—When this option is in enabled state, any Announce message received on the PTP Port in SLAVE or UNCALIBRATED state is accepted and used to update the relevant data sets per 17.6.5.4. If Announce messages from more than one PTP Instance are received, thrashing can occur.

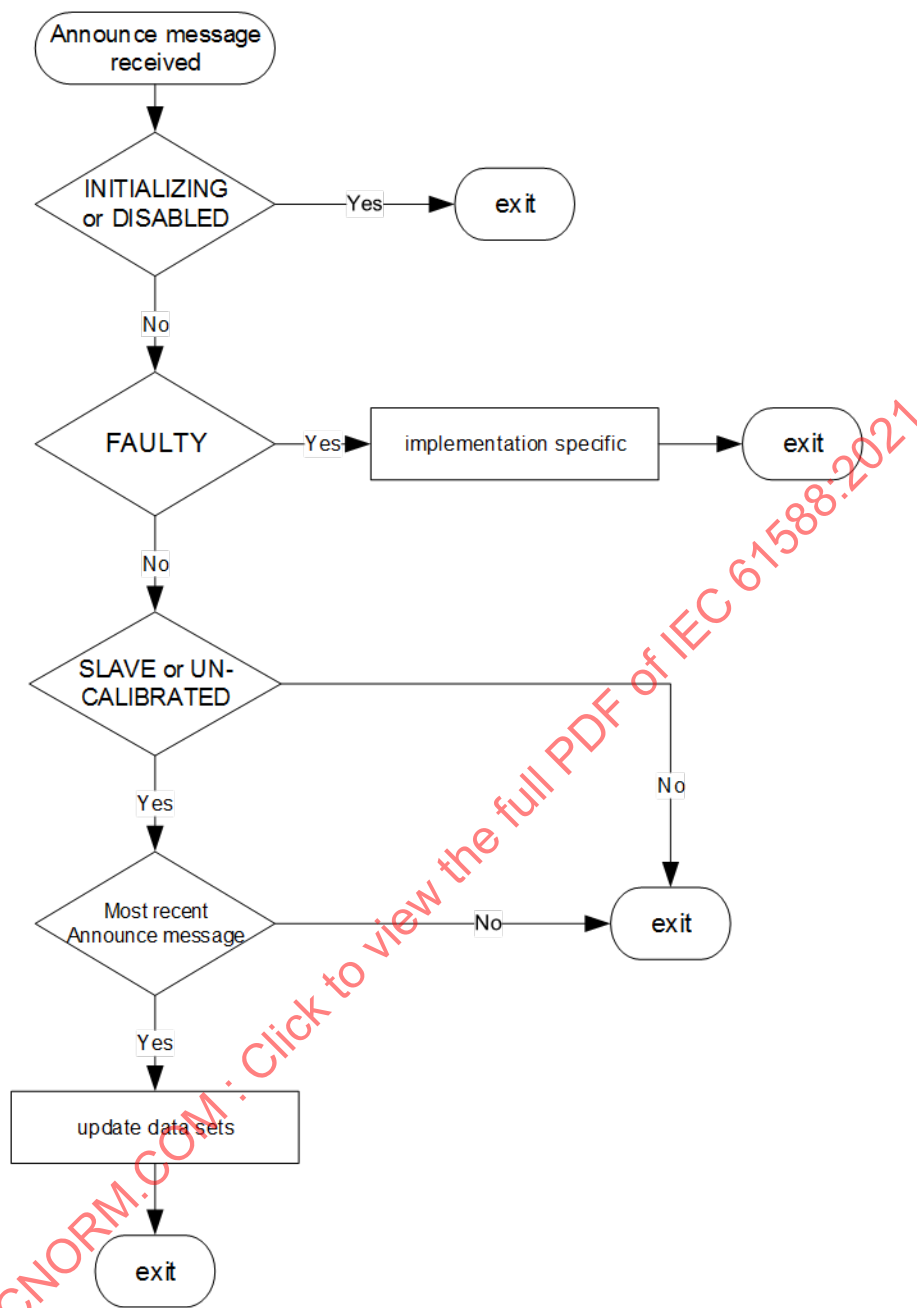


Figure 54—Receipt of an Announce message when the value of the attribute defaultDS.externalPortConfigurationEnabled is TRUE

17.7 Reduced state sets and use of the <foreignMasterList> feature (optional)

17.7.1 General specifications

When this option is implemented (see 6.1), by being specified in the applicable PTP Profile or by the manufacturer, then the option shall operate as specified in this subclause.

This optional feature permits PTP Profiles or manufacturers to specify operation using a reduced set of the current PTP Port states defined in Table 27, and to specify the use or nonuse of the <foreignMasterList> (see 9.3.2.4).

17.7.2 Optional use of selected PTP Port states

The design-specific attributes PD, PF, PL, PP, and PU shall determine the transition logic of Figure 30 and Figure 31 in 9.2.5. Unless specified otherwise in the applicable PTP Profile, the values shall be TRUE. A profile may designate any or all of these values to be FALSE. The result of designating a FALSE value is shown in Table 138.

Table 138—Meaning of FALSE option values

Value	Result
PD	The DISABLED state is not used
PF	The FAULTY state is not used
PL	The LISTENING state is not used
PP	The PRE-MASTER state is not used
PU	The UNCALIBRATED state is not used

17.7.3 Optional use of the <foreignMasterList> feature

The applicable PTP Profile may designate that by design, the <foreignMasterList> feature of 9.3.2.4 is not to be used. The design-specific attribute FMD determines the transition logic of Figure 36 in 9.5.3 and whether the <foreignMasterList> feature is to be used.

Unless specified otherwise in the applicable PTP Profile, the value of FMD shall be TRUE; in which case, the <foreignMasterList> feature is used. A PTP Profile may designate this value to be FALSE indicating that the foreign master feature is not to be used.

18. Interactions between PTP Instances in different PTP domains

18.1 General specifications

The operation of the PTP protocol within a domain is specified by the other clauses of this standard.

This clause includes the following:

- A listing and discussion of the interfaces provided by this standard to enable interdomain interactions
- Conditions that need to be satisfied by interactions between PTP Domains
- Options specifying interactions involving more than a single PTP Domain

NOTE—The interfaces referenced in item a) are primarily intended to enable optimal deployment of a single-domain PTP Network. However, they also enable interdomain interactions as discussed in this clause.

Example applications of interactions between PTP Instances in different PTP domains based on the terms of this clause are considered in informative Annex O.

18.2 Interfaces enabling interdomain interactions

Interactions between multiple PTP domains shall be established using interfaces into and out of a domain specified in this standard. The interfaces include the following:

- a) The Source Dependent block of the layered architecture (see Figure 5, Figure 6, and 7.6.6). Profile specific information may be included in the meta-data provided by the Source Adapter block, see 6.5.2.1.2.

NOTE 1—The Source Dependent Block allows the introduction of timing to be distributed within the domain from an external source such as a GNSS receiver or another PTP Domain.

- b) The Sink Dependent block of the layered architecture (see Figure 5, Figure 6, and 7.6.7).

NOTE 2—The Sink Dependent Block allows a domain to provide timing to an external Clock Sink such as a sensor device or another PTP Domain.

- c) The management mechanism specified in 8.1.4.3.

NOTE 3—The management mechanism allows external entities to configure certain attributes and operational features of this standard, for example, the priority1 attribute of a PTP Instance, or the rate at which Announce messages are transmitted.

- d) The mechanisms for external configuration of a PTP Instance's PTP Port state specified in the options of 17.6.

NOTE 4—The external configuration option allows external mechanisms to configure the state of PTP Ports.

- e) The mechanism for providing path delays from a common service specified in 16.6.

19. Compatibility of this edition with earlier and future editions

19.1 General

This clause discusses the compatibility of PTP Instances conformant to this edition with implementations conformant to other editions.

Compatibility considerations are based on observable behavior external to a PTP Instance. It is therefore possible under some circumstances that implementations of the current edition using a different architectural model or internal procedures that are not identical to those of this edition of the standard will nevertheless produce identical observable external behavior. The specifications of 6.1 explicitly acknowledge this phenomenon.

For the purposes of Clause 19, two similarly configured implementations of the PTP protocol are compatible if the substitution of one implementation for the other in a PTP Network does not cause protocol execution or clock synchronization to fail. Similarly configured means that: both implementations are conformant to the same PTP Profile, the same options are enabled, and they share the same values for configurable attributes. This does not rule out compatibility under other circumstances, for example, one implementation enabling an option that does not interfere with the operation of other implementations. By definition, an implementation conformant to the 2008 edition cannot be similarly configured to an implementation conformant to this edition if the latter enables an option not present in the 2008 edition. However, in this case there might be circumstances in which compatibility is achieved (see 19.4.1).

Compatibility does not imply that no changes are required to bring devices conformant to earlier editions into conformance with this edition. For example, there are changes in this edition that correct technical errors in the earlier edition.

19.2 Compatibility between version 2 and future versions

A PTP Instance implementing this edition of the standard that receives a PTP message with a value of the versionPTP field greater than 2 shall not process the received message. PTP messages with equal or different values for the minorVersionPTP field but equal values for the versionPTP field shall be accepted for processing by the protocol.

NOTE 1—The value 2 for the versionPTP field specified in 7.5.4 is the same as the versionPTP field value specified in the 2008 edition. It is expected that future editions that maintain compatibility with the 2008 edition (and thus also with this edition) will also use a value of 2 for the versionPTP field. Future editions not generally compatible with implementations of this edition are expected to use a larger number for the value of the versionPTP field.

NOTE 2—Compatibility with future editions that use a value of 2 for the versionPTP field is expected to be maintained if the PTP Instances are appropriately configured and deployed.

19.3 Compatibility with IEEE Std 1588-2002

Starting with this edition, compatibility with the specifications of IEEE Std 1588-2002 is no longer supported. This does not preclude implementations using the specifications of Clause 18 of the 2008 edition and the suggestions of informative Annex O of this edition from interfacing between networks composed of implementation conformant to IEEE Std 1588-2002 and networks composed of PTP Instances conformant to this edition.

A PTP Instance implementing this edition of the standard that receives a PTP message with a value of the versionPTP field of 1 shall not process the received message unless implementation specific means to translate between the specifications of the protocol of the two editions are in effect.

NOTE—The value of the versionPTP field specified in the 2002 edition is 1.

19.4 Compatibility between the PTP Instance conformant to this edition and the implementations conformant to IEEE Std 1588-2008

19.4.1 General

PTP Instances conformant to this edition and implementations conformant to the 2008 edition are compatible (see 19.1), provided:

- None of the options of either edition are used, or
- Only options present in both the 2008 and the current editions are enabled and similarly configured.

PTP Instances conformant to this edition and implementations conformant to the 2008 edition in some cases can be compatible even when the implementation conformant to this edition enables an option not present in the 2008 edition (see Table 139 for cautions for this circumstance).

Compatibility and cautions when options of this and the 2008 edition are used in the same PTP Network are summarized in Table 139. In this table “new implementations” refer to PTP Instances conformant to this edition and “old implementations” refers to implementations conformant to the 2008 edition. N/A indicates that the option was not present in the 2008 edition.

Table 139—Compatibility of options

Option	2008 subclause	This edition subclause or annex	Compatible	Cautions
Unicast message negotiation	16.1	16.1	YES	
Path trace	16.2	16.2	YES	
Alternate timescales	16.3	16.3	YES	Improvement if only new implementation (see 19.4.4).
Holdover upgrade	N/A	16.4	YES	
Profile isolation	N/A	16.5	YES	If the sdoId of both new implementation option and old implementations are identical both sdoIds are either 000 ₁₆ or implementations are compatible (see 19.4 implementation uses a different sdoId process PTP messages from old implementation by definition the new implementation use profile than the old implementation (see 16
Common Mean Link Delay Service	N/A	16.6	YES	An old implementation present in Communication Path between implementations will prevent this option from working correctly (see 19.4.13).
Configurable correction of generated timestamps	N/A	16.7	YES	
Calculation of delay asymmetry	N/A	16.8	YES	
Mixed multicast unicast operation	N/A	16.9	YES	Old implementations might require negotiate unicast contract for compatibility (see 16.9
Cumulative frequency transfer	N/A	16.10	NO	An old implementation receiving timing information using this feature will not use the CUMULATIVE_RATE_RATIO TLV part of the timing information. An old implementation present in a PTP Communication Path will prevent new implementations from working correctly.
Slave event monitoring	N/A	16.11	YES	
Enhanced synchronization accuracy metrics	N/A	16.12	YES	Only new edition PTP Instances which implement this feature will be able to provide metrics.
Message length extension	N/A	16.13	YES	
PTP integrated security mechanism	N/A	16.14	NO	Some combinations of network topology and policies might prevent operation in either old and new edition compliant implementations.
Grandmaster clusters	17.3	17.2	YES	
Alternate master	17.4	17.3	YES	
Unicast discovery	17.5	17.4	YES	
Acceptable master table	17.6	17.5	YES	Old implementations do not work if Transports are present (see 19.4.4).
External configuration of PTP Port state	N/A	17.6	YES	External port configuration works only when states are carefully chosen with respect to topology.
Reduced states and foreign master feature	N/A	17.7	YES	
Performance monitoring	N/A	Annex J	YES	
Layer 1 based synchronization performance enhancement	N/A	Annex L	YES	Performance improvements can be suboptimal if implementations are present (see 19.4.11).

The remaining subclauses discuss in more detail caution entries in Table 139 as well as clarification of compatibility resulting from other changes to the standard including the following:

- a) Changes to correct errors in the 2008 edition: These include both technical errors as well as typographical and reference errors.
- b) Changes required by changes in the policies of the IEEE Registration Authority²² that impact the assignment of clockIdentity values.
- c) Changes to improve clarity: These include changes arising from interpretations given prior to the discontinuation of interpretations activity by IEEE-SA.
- d) Changes to allow more flexible implementation of existing specifications.

19.4.2 Revision of clause 6- “Clock synchronization model”

In both the 2008 and this edition, Clause 6 presents the architectural model underlying the specifications, describes certain aspects of the behavior of the protocol, and outlines the structure of the specifications.

The first major change in Clause 6 is the replacement of the 2008 architectural model by a layered model. The new model permits the following:

- Introduction of capabilities enabled by a new PTP Port model: The Special Port. Special Ports enable the inclusion of network links based on technologies that provide inherent timing support as opposed to the use of PTP timing messages, for example, IEEE Std 802.11 and EPON (see 19.4.6)
- Principled interactions between different PTP domains

The second major change is the removal of discussions of different models of clock synchronization, material that was never referenced in the 2008 edition.

These changes do not affect the compatibility of implementations conformant to the 2008 and this edition.

19.4.3 Clarification of terms

While the 2008 edition acknowledged PTP Networks supporting several PTP domains, the specification was primarily directed to single-domain systems. As a consequence, the terminology used in the 2008 edition was in many respects inconsistent and ill-defined when applied to multidomain PTP Networks. In this edition, the definitions in Clause 3 and a revision of 6.1 correct this situation.

These changes do not affect the compatibility of implementations conformant to the 2008 and this edition.

19.4.4 Updates to options present in 2008 edition

The following summarizes changes that affect compatibility for option present in the 2008 edition (see Table 139).

- **Alternate timescales (see 16.3):** The changes include the clarification of the propagation of the ALTERNATE_TIME_OFFSET_INDICATOR TLV by Boundary Clocks, and clarification on using this option. PTP Instances conformant to this edition are compatible with those of the 2008 edition.
- **Acceptable master table (see 17.5):** The representation of acceptable masters has been changed from protocol address to portIdentity to correct a possible flaw in the 2008 specifications, relevant

²² IEEE Registration Authority (<https://standards.ieee.org/regauth/>).

when Transparent Clocks are part of the PTP Network topology. PTP Instances conformant to the 2008 edition, within a network including Transparent Clocks that update the source address per specification in IEEE 802.1, might incorrectly not accept messages from a configured acceptable master. PTP Instances conformant to this edition will correctly accept the message irrespective of whether Transparent Clocks are present.

19.4.5 Revision of the specifications of a Transparent Clock

Two major changes have been made in the specifications of Transparent Clocks. The optional data sets for Transparent Clocks in the 2008 edition have been replaced by attributes in the defaultDS and portDS data sets and the specifications are now explicitly per PTP Instance and therefore domain specific.

In the 2008 edition, the specifications were domain independent at least concerning the data sets (see the NOTE in 8.3.1 of the 2008 edition concerning the optional data sets). PTP Networks containing implementations conformant to the 2008 edition might not operate correctly in networks including PTP Instances of this edition if the following occur:

- The PTP Instances of this edition require a Transparent Clock to modify a PTP message field other than the correctionField (and any resulting updates of transport layer fields), and/or
- Options of this edition or of PTP Profiles have specifications that are explicitly per domain.

19.4.6 Special Ports

Special Ports are not specified in the 2008 edition. A Special Port must be connected directly to another compatible Special Port. Therefore, PTP Instances with a Special Port cannot be used in a network unless 1) the Special Port is not connected to the network or 2) the Special Port is directly connected to another Special Port.

19.4.7 Revised specifications for domains

The specifications defining a domain in this edition include two fields, the domainNumber and the sdoId. The structure of the sdoId is such that compatibility with the 2008 edition is maintained. (A portion of the sdoId is the repurposed transportSpecific field from 2008.) The compatibility of the sdoId with the corresponding attributes of the 2008 edition ensures that the Profile Isolation option of 16.5 correctly operates in the presence of implementations conformant to the 2008 edition.

19.4.8 New options permitting management configuration of the states of PTP Ports

The new options, 17.6 and 17.7, permit external configuration of PTP Port state in PTP Instances conformant to this edition. These changes are compatible with the specifications of the 2008 edition. The presence of implementations conformant to the 2008 edition in a PTP Network of PTP Instances conformant to this edition can limit the utility of these new options in controlling PTP Network topology since the port states of 2008 devices are determined by the BMCA and not by configuration.

19.4.9 New rules on values for clockIdentity

Several of the specifications for clockIdentity values permitted under the 2008 edition are no longer permissible due to the small but finite possibility of duplication. clockIdentity values assigned based on this edition are compatible with those of the 2008 edition. However, the presence of implementations using the specifications of the 2008 edition does introduce the possibility of duplicate clockIdentities in a PTP Network (see NOTE 2— of 7.5.2.2.2.1 of this edition).

19.4.10 Revision of Clause 8

The data set specifications of Clause 8 are revised and are now to be interpreted as follows:

- An information model to be used as the basis for constructing management tools. For example, management tools based on a MIB [B26] or YANG [B22] model can be constructed from the specifications of Clause 8, and/or
- A definition of data used in the operation of the PTP protocol.

As part of this revision, the existing management specifications, Clause 15 of the 2008 edition, are revised such that they are based on the revised data sets, and any existing management specifications present in options of the 2008 edition are now specified in Clause 15 of this edition. The use of Clause 15 specifications of this edition is compatible with those of the 2008 edition.

19.4.11 New options and default profile for enhanced synchronization performance

The new options are defined in 16.7, 16.8, and Annex L. The new default profile is defined in I.5. PTP Instances using these options and/or profile will operate correctly in a PTP Network of implementations conformant to the 2008 edition. Since 2008 devices do not implement these options, the improvement of synchronization accuracy will be sub-optimal compared to a PTP Network where all PTP Instance implement these options.

19.4.12 Option for using cumulative rate ratios

The operation of this option with the specifications of the 2008 edition will result in synchronization errors if a PTP Instance conformant to this edition with this option enabled is in the PTP Communication Path of implementations conformant to the 2008 edition. These errors are due to incorrect computation of <meanLinkDelay> by PTP Instance using this option due to failure to receive the Cumulative_Rate_Ratio TLV from a 2008 implementation.

19.4.13 Optional Common Mean Link Delay Service providing <cumulativeRateRatio>, <neighborRateRatio> and path delays in multi-domain PTP networks using the peer-to-peer delay mechanism

This option is not useful with implementations conformant to the 2008 edition but will not disrupt the operation of the protocol in a PTP Network of such implementations.

19.4.14 Options for greater security

Implementations of items in informative Annex P on security are generally applicable to PTP Networks containing implementations conformant to the 2008 edition. However, the use of the AUTHENTICATION TLV (see 16.14.3) will disrupt the operation of implementations not using this TLV, including implementations conformant to the 2008 edition.

20. Conformance

20.1 Conformance objective

The philosophy underlying the conformance requirements of this clause is to:

- Raise the level of interoperability of systems built to this standard.
- Encourage the manufacture of PTP components with the broadest possible range of applicability.
- Provide opportunity for continued technical improvement and differentiation.

20.2 PTP conformance requirements

20.2.1 General conformance specification

Conformance requirements are specified in terms of PTP Instances.

PTP Instances shall conform to all clauses of this standard with the exception of:

- Clauses specifically marked “optional” unless mandated by the PTP Profile in use.
- For applications that distribute only frequency and do not require the measurement of the path delays, an alternate PTP Profile may specify that the path delay mechanisms of 11.3 and 11.4 shall not be implemented or activated.
- There are situations where a PTP Instance participates in time transfer but the accuracy requirements are such that, for a portion of the PTP Network path including the PTP Instance, path delays can be neglected. In this case a PTP Profile may specify that the mechanisms of 11.3 and 11.4 for measuring path delays are inactive by configuration of the value of `portDS.delayMechanism` (see 8.2.15.4.4) to `NO_MECHANISM`.

For each option implemented, the PTP Instance shall conform to the clause specifying the option.

20.2.2 Transport conformance specification

A PTP Instance that uses a transport protocol for which the mapping is defined in an annex of this standard shall conform to that annex.

The transport of PTP messages using a transport protocol for which there is no mapping defined in this standard shall be defined by a mapping defined and published by the standards organization, or its designee, with jurisdiction over the transport. The publication specifying this mapping shall be referenced by the applicable PTP Profile.

20.2.3 PTP Profile conformance specification

A PTP Instance claiming compliance shall specify at least one PTP Profile to which it complies. One of the three default PTP Profiles shall be used in the absence of a suitable alternate PTP Profile. The default PTP Profiles are specified in Annex I.

If a particular attribute or option is not specified in the selected PTP Profile, then the PTP Instance shall conform to the value or choice specified in the default PTP Profile that specifies the same path delay mechanism.

All PTP devices should support at least one of the default PTP Profiles.

20.3 PTP Profiles

20.3.1.1 General

The purpose of a PTP Profile is to allow organizations to specify specific selections of attribute values and optional features of PTP that, when using the same transport protocol, interwork and achieve a performance that meets the requirements of a particular application.

A PTP Profile is a set of required options, prohibited options, and the ranges and defaults of configurable attributes. Profiles specifications shall be consistent with the specifications in 20.2.1 and 20.2.2.

20.3.1.2 PTP Profile recommendations

A PTP Profile should define the following:

- Which of the best master clock algorithm options (see 9.3.1) is to be implemented
- Which of the configuration management mechanisms (see 8.1.4.3) is to be implemented
- Which of the path delay mechanisms, delay request-response (see 11.3) or peer-to-peer delay (see 11.4) is to be implemented
- The range and default values of all PTP configurable attributes and data set members subject to the restrictions of 8.1.2.1.3
- The transport mechanisms required, permitted, or prohibited
- The PTP Instance types required, permitted, or prohibited
- The options required, permitted, or prohibited, and any parameter values associated with these options
- Uncertainty specifications appropriate to the evaluation of whether traceability (see 3.1.80), to a primary reference is achieved for time (see 8.2.4.6) and for frequency (see 8.2.4.7)
- The value of the observation interval (τ) used for variance measurement (see 7.6.3.2)

A PTP Profile shall extend the standard only in the following ways:

- a) The use of the TLV mechanism of 14.3
- b) The specification of an optional best master clock algorithm (see 9.3.1)
- c) The specification of an optional management mechanism (see 8.1.4.3)
- d) The provisions of 20.2.2
- e) The provisions of 7.3

NOTE—There are many places in the standard where it specifies the use of values or alternatives designated for PTP Profiles. This is not considered an extension of the standard. For example, clockClass values (see 7.6.2.5) in the range 68 through 122 are indicated for use by alternate PTP Profiles.

20.3.2 Specific PTP Profiles

A PTP Profile may be developed by external organizations, including:

- a) A recognized standards organization with standards activity relevant to an industry, for example, the IEC, IEEE, IETF, ANSI, or ITU, or
- b) An industry trade association, regulatory, government, or other similar organization writing standards for an industry, or
- c) Other organizations as appropriate.

Standards development organizations in the first two categories above are defined as QSDOs for the purposes of option 16.5 provided:

- d) The organization's membership consists of representatives from multiple companies or academic or government entities who are active within the industry governed by the organization. The organization must not be dominated by representatives of a single entity where dominance is "defined as the exercise of authority, leadership, or influence by reason of superior leverage, strength, or representation to the exclusion of fair and equitable consideration of other viewpoints" (IEEE Standards Board Bylaws section 5.2.1.3).
- e) The PTP Profiles developed by the organization must be approved in one of the following ways:
 - 1) By a vote of the membership according to the organizations bylaws if such exist.
 - 2) Based on the normal operation procedures agreed upon by the organization.

NOTE—Option 16.5 allows QSDOs the option of ensuring that PTP Profiles written by that organization are isolated from PTP Profiles of other organizations. Isolation from other PTP Profiles of their own organization can be achieved by PTP Profile specification of restrictions on the domainNumber attribute within the allowed domainNumber range for their sdoId. QSDOs can also specify PTP Profile operation with an sdoId value of 000₁₆, essentially the space open to all as it was in the 2008 edition.

Non-QSDOs must specify PTP Profile operation with an sdoId value of 000₁₆, essentially the space open to all as it was in the 2008 edition. QSDOs writing a PTP Profile under the terms of 16.5 must obtain a value for their sdoId from the IEEE Registration Authority.

Only a single sdoId will be issued to each QSDO by the IEEE Registration Authority. If a QSDO wishes to write multiple PTP Profiles isolated from each other under the terms of 16.5, it must do so based on other mechanisms such as a required TLV or further restriction of the domainNumber space.

The PTP Profile development organization should consult the IEEE Precise Networked Clock Synchronization Working Group (also known as the "IEEE 1588 Working Group") of the IM/ST Committee for technical review.

20.3.3 PTP Profile specifications

A PTP Profile shall be identified by the following attributes:

- **profileName:** This attribute shall be the text title of the profile as designated by the organization specifying the profile.
- **profileNumber:** This attribute shall be a (UInteger8) number assigned by the organization specifying the profile, and among all profiles whose identifiers use the same OUI or CID [B12], shall be unique to the profile with the title profileName.

NOTE 1—IEEE Std 1588-2008 did not have the concept of profileNumber, but the profileIdentifier scheme in this edition is compatible with 2008.

- **profileVersion:** This attribute shall be the version of the profile as designated by the organization specifying the profile. The version designation shall consist of two fields: A primaryVersion (UInteger8) and a revisionNumber (UInteger8). The profileVersion shall be printed as “Version primaryVersion.revisionNumber”. profileVersion values shall be assigned to distinguish versions and revisions of the profile identified by the profile attribute.
- **profileIdentifier:** This attribute shall consist of 6 octets. The first 3 octets shall be either an OUI or a CID owned by the organization creating the profile. The next octet shall be the profileNumber attribute. The next 2 octets shall be the profileVersion attribute.

NOTE 2—If the organization specifying the profile already owns or purchases an OUI, it can be used in the profileIdentifier. As of January 2014, IEEE RA rules governing OUIs restrict the sales of OUIs under certain circumstances. Therefore, a CID obtainable from the IEEE RA can be used in the profileIdentifier.
- **organizationName:** This attribute shall be the textual name of the organization specifying the profile and owning the OUI or CID of the Profile Identifier.
- **sourceIdentification:** This attribute shall be a URL, e-mail, or regular mail address to which inquiries concerning the profile or requests for copies may be sent.

In printed matter, the format shown in Figure 55 shall be used with the named attributes as described in this subclause.

PTP Profile: profileName profileNumber profileVersion profileIdentifier This profile is specified by the organizationName A copy may be obtained from sourceIdentification

Figure 55—Profile print form

Annex A

(informative)

Using the Precision Time Protocol (PTP)

A.1 Overview

PTP provides a simple methodology for accurately synchronizing PTP Instances in a network. When designing such a network, the following questions need to be answered:

Physical layout issues:

- How physically dispersed are the PTP Instances?
- What network technology is to be used?

Logical issues:

- Is the system a single collection of PTP Instances, or are the PTP Instances divided into logical groupings each with their own sense of time?

Component issues:

- How accurately do the PTP Instances need to be synchronized?
- What is the source of time for the system? Does it need to be traceable to UTC or TAI?

Local implementation issues:

- How are timing requirements to be met?
- How do other applications sharing the communication network affect PTP?
- How do accuracy requirements affect the implementation?
- What are the design issues for local oscillators?

System implementation issues:

- How is the system partitioned?
- Which options are used?
- Which profiles are used?

Performance issues:

- How do network delays and fluctuations affect PTP Instance Time accuracy?
- How does clock oscillator stability affect PTP Instance Time accuracy?

Conformance testing issues:

- What features aid in conformance and performance testing?
- What features aid in calibrating PTP Instance timing?

Subclause A.2 through subclause A.9 address each of these topics.

A.2 Physical layout

PTP Instances communicate with each other over a PTP Network. Typically, the selection of the network technology is based on the primary application. PTP works on any message-based network. PTP is designed to work in a multicast environment, although it is possible to design unicast PTP components and networks. Ethernet is a well-suited network for implementing PTP, and the rest of this annex uses Ethernet as an example.

All networks have limitations on distance, number of allowed PTP Instances, and traffic. If the PTP Instances to be synchronized are dispersed beyond the range of the network technology, then the networks are preferably designed as separate “islands of time” with provision outside of PTP for synchronizing these islands.

For example, if the network consists of two compact sites separated by several miles, PTP can be used within each site, with site-to-site synchronization provided by another technology such as GPS.

Within a site, distance, traffic, and number of PTP Instance issues are usually addressed by special network components. For Ethernet, localized devices typically communicate via bridges. For larger and more complex networks, routers are used to separate the network into regions using only bridges. In general, each level of separation using these devices introduces additional statistical delay and delay fluctuation in the PTP message transmission times between devices.

PTP is designed to minimize the effects of delay and delay fluctuation. To get the best PTP performance, the PTP Network topology preferably minimizes the number of such separating devices between PTP Instances with the most critical synchronization requirements.

Bridges and routers not implementing PTP can introduce considerable timing jitter and path asymmetry. Although such devices can be included in a PTP Network, use of these devices is not advisable unless timing errors introduced by their jitter and path asymmetry are tolerable for the application, or can be reduced by an appropriate filtering algorithm. These effects can be mitigated by implementing PTP in bridges and routers (i.e., Boundary Clocks and/or Transparent Clocks).

A.3 Logical layout

Most applications consist of a single set of PTP Instances to be synchronized. For this case, all the PTP Instances can be placed in a single domain. If the default values specified in this standard and the applicable conformant PTP Profile are used, then generally no configuration of the PTP Instances is necessary.

If the application requires several groups of PTP Instances, with each group maintaining a different self-consistent timescale, then one of the following two solutions can be used:

- If the rest of the application is segmented into the same groups, it might be possible to use separate noncommunicating PTP Networks; in which case, each group can use the default domain. Network routers are often used for this purpose.
- If the groups have to share a common PTP Network, then each group can be assigned to a different domain. This logically divides the PTP Instances as desired. Depending on the mapping to the underlying physical addressing of the network, the processing load on each PTP Instance might or might not be affected.

With the exception of the assignment of PTP Instances to a domain, PTP defines an administration-free network in the default case. Within a domain, PTP Instances can be added or removed without any requirement for modification of address tables, etc., provided components use the recommended multicast communication model. Addition or removal of PTP Instances might cause a different PTP Instance to become the Grandmaster PTP Instance in the network. This might cause a transient in the timescale as the network automatically recalibrates for the new delay patterns to the new Grandmaster PTP Instance.

This standard provides several configuration options for users that require more control over the selection of Master PTP Instances, or over different timing and other attributes that govern network performance. For example, the use of the priority1 attribute allows network designers to designate up to 254 devices in a priority order for Grandmaster PTP Instance selection.

A.4 Component issues

The primary issue in the selection of PTP Network components is the required synchronization accuracy.

- PTP Instances that support protocol features protocol required to achieve the desired accuracy need to be selected.
- Network components and physical design decisions also affect the accuracy as outlined in the previous clauses.

Properly designed Ethernet PTP Networks can readily achieve sub-microsecond accuracy.

A second issue is the technique for establishing the PTP Network epoch. In every domain, the epoch is defined by the Grandmaster PTP Instance that is selected according to the best master clock algorithm.

If TAI or UTC traceable time is a requirement, then the Grandmaster Clock maintains the timescale PTP.

If the value of clockClass is 6, 7, 52, or 187 for the Grandmaster PTP Instance in a domain, the timescale is PTP. From the timescale PTP, UTC can be computed using the value of currentUtcOffset distributed by PTP. Such networks might or might not maintain the epoch after a power outage (see 7.6.2.5).

If the value of clockClass is 13, 14, 58, 193, or 216 or greater for the Grandmaster PTP Instance in a domain, the timescale is ARB (see 7.6.2.5). Such networks might or might not maintain the epoch after a power outage.

A Master PTP Instance can fail in such a way that its time or frequency become incorrect. Detection of this problem and recovery from it, within a single domain, are outside the scope of this standard. Some information such as the Parent PTP Instance statistics maintained in the parentDS data set is available to aid in detecting a “false-ticking” master. Implementers are advised to consider information from as many PTP Instances as possible, and to weigh the information from each PTP Instance according to the inherent stability of the Local PTP Clock of the PTP Instance. A PTP Network containing three or more domains using the same time reference can identify false ticking masters by using a voting algorithm (see O.3).

Disabling a PTP Port with management (i.e., write portDS.portEnable to FALSE) can aid in recovering from a false-ticking master. Note that disabling or demoting a Master PTP Instance has side effects (especially if it is a Boundary Clock), so the decision to do that might depend on factors besides its timekeeping quality. That decision is outside the scope of this standard.

A.5 Local implementation issues

A.5.1 General

This subclause provides some guidelines for implementers of PTP Ordinary Clocks, Boundary Clocks, and Transparent Clocks. Although not within the scope of this standard, note that services built on top of PTP Instances synchronized via PTP (or any other protocol) might degrade the accuracy beyond acceptable limits.

A.5.2 Timing issues

Implementations need to meet the PTP message processing and timing requirements and also need to meet whatever timing requirements are needed to operate any servomechanism that synchronizes the Local PTP Clock based on information in PTP messages.

Implementations need to ensure that adequate computing and memory resources are available to meet these requirements. Implementations also need to ensure that the resources needed by the PTP implementation have adequate priority over other applications sharing these resources to meet the PTP and servomechanism timing requirements. PTP tasks are preferably assigned the highest priority in an implementation, similar to priorities assigned to the protocol stack and other operating network resources.

PTP implementations normally require resources for a short time in every syncInterval. The selection of the syncInterval for a network needs to be consistent with the available resources in all network components.

The use of PTP Network resources by other applications can affect PTP accuracy as discussed in A.5.3.

A.5.3 Accuracy issues

A.5.3.1 General

The achievable accuracy of a PTP Network is limited by the following:

- Delay fluctuation in the protocol stacks of PTP Instances
- Delay asymmetry
- Delay fluctuation in network components
- Timestamping accuracy
- Stability issues

A.5.3.2 Protocol stack delay fluctuation

The simplest implementations of PTP operate as ordinary applications at the top of the network protocol stack. Timestamps are generated at the application level. Protocol stack delay fluctuation causes errors in these timestamps. These errors are typically in the hundred microseconds to milliseconds range, depending on the operating system.

Implementations might generate timestamps at the interrupt level rather than at the application level. In this case, delay fluctuation typically can be reduced to tens of microseconds depending on other use of interrupts by other applications, and on the traffic patterns on the network.

A.5.3.3 Network component delay fluctuation

Network components introduce fluctuation in the propagation time of PTP messages. This directly affects the accuracy of the `currentDS.offsetFromMaster`, `currentDS.meanDelay`, and `portDS.meanLinkDelay` values.

Network bridges and routers are subject to store-and-forward delay fluctuation. Typical Ethernet bridges have input and output buffers communicating over a very high-speed back plane or switch fabric. Each PTP Port typically connects directly to an end device or another Ethernet bridge. The dominant contribution to delay fluctuation arises from the output buffering and queuing. If the output subnet is always available, this delay fluctuation is typically in the nanoseconds range and reducible by averaging techniques. Intensive traffic in a network might cause increased delay fluctuation due to this output buffering. This increased delay fluctuation is much more difficult to reduce. The proper design of PTP Networks need to recognize this effect and take measures to reduce the impact.

Most bridges and routers support traffic prioritization. High-priority traffic suffers less fluctuation in propagation time. PTP event messages preferably are sent with high priority compared with other data whenever possible. See Annex D through Annex I for specific priority recommendations for each transport protocol.

A.5.4 Timestamp accuracy

The resolution of the clock generating the timestamps required by PTP needs to be consistent with the desired accuracy. Note that this resolution contributes to the PTP variance (see 7.6.3).

A.5.5 Stability issues

As noted earlier in this annex, the delay fluctuation introduced into the computation of the `currentDS.offsetFromMaster` and `currentDS.meanDelay` members can be reduced by suitable design of any synchronization servo algorithms of the PTP Instance. Engineering trade-offs need to be made between the averaging times (number of samples) and the responsiveness to effects other than delay fluctuation, such as oscillator stability.

The fundamental time stability of the Local PTP Clock needs to be consistent with the required `syncInterval` and accuracy specifications. The algorithms used to reduce delay fluctuation do not correct for drifts of the local clocks during time intervals that are small compared with the averaging intervals of the algorithms. Servos cannot correct for random drifts occurring within a `syncInterval`.

At high accuracy, the specifications on the stability of the local oscillators driving the Local PTP Clock can be quite difficult to meet. The trade-off is between cost and stability. Local oscillators typically are quartz crystals. The frequency of quartz crystals typically drifts due to thermal, mechanical, and aging effects. Of these effects, thermal ones are the most difficult to deal with in most applications.

For example, a typical thermal specification for uncompensated crystals is 1 ppm per degree Celsius. A 1° temperature rise over a `syncInterval` of 2 s produces an error on the order of 2 μs. Accuracies in the tens of nanosecond range, therefore, imply that some combination of better thermal specifications on the crystal, reduced `syncInterval`, and better thermal management be used to reduce the thermal drift by two orders of magnitude.

PTP allows `syncInterval` to be reduced to a fraction of a second depending on the PTP Profile selected, with the corresponding increase in computation and network bandwidth requirements.

Thermal specifications on crystals become increasingly expensive below 1 ppm/degree. Control of the thermal environment needs to be carefully managed, particularly in high accuracy implementations. Very long averaging times typically require oven-controlled crystals or the use of more stable oscillators. Thermal drift during the short intervals and averaging times typical of PTP Networks can often be managed by attention to heat dissipation in surrounding devices, cooling patterns within the device, increasing the thermal mass of the oscillator, and similar techniques. See Sullivan et al. [B52] for a thorough discussion of clock characterization.

A.6 System implementation issues

A PTP Network is the collection of PTP components that operate together to meet the requirements of an application. An interoperable PTP Network is one where the protocol operates as specified in this standard, the selection and configuration of PTP Instances is such that the protocol is successful in constructing a master–slave timing hierarchy, and the PTP Instances with a PTP Port in the SLAVE state are able to synchronize to Master PTP Instances. An optimal PTP Network is one that is interoperable, manageable, and meets the synchronization requirements of the application. Ensuring that a network built with conformant PTP Instances is optimal is an issue for the network integrator. The following recommendations facilitate the construction of interoperable networks:

- Use a single transport media (e.g., Annex C to Annex H) throughout the domain, or divide the domain into regions, each of which uses a single transport. Regions are connected using Boundary Clocks.
- Use a single management approach throughout the network. Either the PTP management message mechanism of this standard or an alternate management mechanism specified in a PTP Profile are acceptable.
- Use the same choice of best master clock algorithm throughout the domain. There is no assurance that regions of a domain implementing different choices of best master clock algorithm can be made to interoperate, even when connected by a Boundary Clock. Use either the best master clock algorithm defined in this standard or an alternate specified in a PTP Profile.
- Use the same selection of state configuration options (Clause 17) throughout the domain. If state configuration options are used, it is the responsibility of the network integrator to ensure that the selected configuration produces an interoperable network. There is no assurance that regions of a domain implementing different choices of configuration options and configuration can be made to interoperate, even when connected by a Boundary Clock.
- Use a single path delay mechanism (see 11.3 and 11.4) throughout the domain, or divide the domain into regions with each using a single path delay mechanism. Regions are connected using one or more Boundary Clocks.
- Use an interoperable set of attribute and configurable data set values throughout the domain, or divide the domain into regions with each using a single such interoperable set. Regions are connected using one or more Boundary Clocks.
- Use the same default value for each attribute and configurable data set member on all PTP Instances in the network.
- Use the same required maximum and required minimum range values for each attribute on all PTP Instances in the network.
- Some options only work as designed, without interoperability problems, if they are implemented and active on every PTP Instance in a network. An example is the security option (see 16.14). Other options are effective on the subset of PTP Instances implementing them, even if other PTP Instances in the network do not support the option. Furthermore, the presence of such PTP Instances does not interfere with PTP Instances not implementing the option. An example is the unicast option.
- Use only PTP Instances implementing the same PTP Profile throughout the domain, or divide the domain into regions, with each using a single PTP Profile. Regions are connected using a Boundary

Clock capable of resolving the PTP Profile differences. There is no assurance that the specifications of two PTP Profiles allow the design of a Boundary Clock that resolves their differences. For example, it is not possible to ensure that regions using self-configuration with the best master clock algorithm of this standard can interoperate with regions that use configuration of the master–slave hierarchy.

- Use only PTP Instances implementing the same version of this standard throughout the domain, or divide the domain into regions with each using a single version (version 1, version 2, or a future version). Connect these regions using a Boundary Clock.

A.7 Guidelines to achieve optimal performance

The following requirements will aid in achieving optimal clock synchronization performance:

- a) The delay of a PTP message exchanged the Master PTP Instance and a Slave PTP Instance need to be symmetric.
- b) A PTP Instance might contain asymmetric delays in its timestamping mechanism or protocol path. If these asymmetries are not negligible, they need to be correctly accounted for (see 7.4.2).
- c) The delay of a PTP message exchanged between a Master PTP Instance and a Slave PTP Instance need to be constant over the time interval between PTP event messages.
- d) Delay fluctuation due to network components and due to the protocol stack within PTP Instances are preferably reduced by two techniques:
 - 1) The timestamps used in PTP need to be generated as close to the physical layer as practical for a given PTP Instance implementation. In cases where the most accurate timestamps can be generated only after a message has been transmitted, the actual value is communicated in a Follow_Up message or a Pdelay_Resp_Follow_Up message.
NOTE—See Eidson et al. [B6], IETF RFC 1589 [B13], and IETF RFC 2783 [B14] for mechanisms to aid in generating these timestamps.
 - 2) Remaining delay fluctuation introduced by the protocol stack and by network components not isolated by a Boundary Clock or Transparent Clock can be reduced by averaging. The averaging algorithms are outside the scope of this standard.
- e) The computing power of PTP Instances implementing the protocol needs to be great enough and the number of PTP Instances needs to be small enough to meet the timing constraints. Implementers of Boundary Clocks and Ordinary Clocks, for example, need to consider the resources required to process Delay_Req messages from Slave PTP Instances communicating with the PTP Instance. The inability to process these messages due to resource limitations can lead to deterioration in the synchronization performance due to missed measurements of the path delays. Users need to be aware of this limitation when selecting PTP Nodes and designing their network.
- f) The inherent stability and precision of a PTP Instance's oscillator need to be adequate; see A.5.4 and A.5.5.

A.8 Recommendations to aid in conformance testing

To aid in 1) testing the performance of a PTP Network, 2) calibrating PTP Instances, and 3) verifying conformance, all PTP Ordinary Clocks and Boundary Clocks preferably provide a 1 pulse per second (PPS) signal with the rising edge of the pulse coincident with each increment in the seconds field of the PTP Instance. If implemented and not coincident, then the PTP Instance specifications preferably include the time offset of the 1 PPS signal from the seconds increment event time. This signal can be an accessible internal test point and need not be visible as an external output of the PTP Node in which the PTP Instance is embedded, for example, a sensor.

A.9 Recommendation for implementations in unicast networks or networks with non-PTP bridges and routers

A.9.1 General

Master PTP Instances and Slave PTP Instances will be introduced into networks where bridges and routers do not support the PTP standard. Furthermore, many networks do not support multicast.

The unicast communication model can be used to overcome many of these problems. Subclause 7.3.1 allows the use of a unicast model provided that the behavior of the protocol is preserved.

This subclause describes issues that need to be resolved in an alternate PTP Profile when using a unicast communication model in order to produce an implementation that is likely to work in such networks, while satisfying a wide range of timing requirements. Some wide-area network requirements, such as security and resilience, are outside of the scope of this discussion.

A.9.2 Boundary Clocks and Transparent Clocks in a unicast model

In the multicast model, Ordinary Clocks and Boundary Clocks create a synchronization hierarchy without prior knowledge of PTP Network topology. Except for PTP management messages, Boundary Clocks terminate all PTP messages. Furthermore, if only PTP bridges, routers, Transparent Clocks, and Boundary Clocks are present, the messages used by the peer-to-peer delay mechanism terminate in the neighbor peer-to-peer PTP Instance. In the unicast model, however, the above conditions do not hold.

To preserve the protocol behavior, the following functions need to be preserved when using the unicast model:

- The correct creation of the synchronization hierarchy.
- The correct exchange of PTP event messages and associated PTP general messages needed for synchronization.
- The correct operation of the peer-to-peer delay or delay request-response mechanisms for determining path latency.
- A management mechanism for configuring the PTP Instances.

One way to achieve these functions is by requiring that all Ordinary Clocks, Boundary Clocks, and peer-to-peer Transparent Clocks be configured in advance with the unicast protocol addresses of the neighboring PTP Instances visible from each PTP Port. As one exception to the previous sentence, the addresses of slave-only PTP Instances using the delay request-response mechanism do not need to be preconfigured in other PTP Instances if unicast option 16.1 is used. If the peer-to-peer delay mechanism is to be used, the configuration needs to ensure that only a single peer-to-peer PTP Instance is visible from each PTP Port (see 11.4.3).

PTP Ports might be neighbors even when there are bridges, routers, or Transparent Clocks between the PTP Ports. PTP Ports are not neighbors if there is a Boundary Clock between them. In the event of a PTP Network reconfiguration, the neighbor relationships can change; in which case, two PTP Ports might communicate in unicast across a Boundary Clock. If a mechanism for learning topology changes is available, PTP Instances can stop all unicast communications between non-neighbors, leading to an optimized synchronization hierarchy and better utilization of the network resources. If such a learning mechanism is not available, then depending on the PTP Network topology, it might be advisable to use end-to-end Transparent Clocks instead of Boundary Clocks or peer-to-peer Transparent Clocks. In all cases, the implementation has to provide a mechanism to break forwarding loops for achieving correct operation of the protocol.

A.9.3 Unicast options

The configuration options of Clause 17 can be used to configure each PTP Port with the needed unicast protocol addresses.

The unicast option of 16.1 can be used to establish unicast communications for Announce, Sync, Delay_Resp, and Pdelay_Resp messages, and any associated messages.

Alternatively, unicast contracts between two PTP Instances can be created using a management mechanism. These contracts consist of the unicast address information and of respective specified packet rates for Sync, Announce, and Delay_Req messages.

The path trace option of 16.2 can be used in defining a mechanism for breaking a loop formed by multiple PTP Communication Paths.

Since the management mechanism of 15.2 depends on the use of the multicast model and on retransmission by Boundary Clocks, an alternative management mechanism for configuring the PTP instances needs to be specified as permitted in 8.1.4.3.

A.9.4 Unicast conformance

A.9.4.1 General

Subclause 20.2.3 specifies that to claim conformance, a PTP Instance needs to comply with a PTP Profile in addition to conforming to the PTP standard. This profile needs to specify any differences from the specifications of the default PTP Profiles of Annex I. Subclause A.9.4.2 contains examples of some of the specifications that are needed to implement a unicast model to meet the requirements discussed in A.9.1 and A.9.2. Not discussed are possible alternate best master clock algorithms and an alternate unicast-based management mechanism.

A.9.4.2 PTP options and attribute values

The unicast options defined in 16.1 and 17.4 need to be supported and operational by default. All other options of Clause 16 and Clause 17 are permitted but need to be inactive by default.

The unicast communication model is used by default as permitted by 7.3.1. If the multicast communication model, or mixed multicast/unicast is also implemented, it needs to be inactive by default. Multicast communication is a recommended option for exploiting future multicast support in these networks and for allowing interoperability with equipment supporting the PTP default profiles.

The timing of unicast PTP messages is determined by the values of the logInterMessagePeriod field in the unicast negotiation REQUEST_UNICAST_TRANSMISSION TLV.

Suggested values for the logInterMessagePeriod field of the REQUEST_UNICAST_TRANSMISSION TLV are as follows:

- **For requesting unicast Announce messages:** The suggested value of logInterMessagePeriod is 1 (once every 2 s). The suggested configurable range is –3 (8 every 1 s) to 3 (once every 8 s).
- **For requesting unicast Sync messages:** The suggested value of logInterMessagePeriod is –4 (16 every 1 s). The suggested configurable range is –7 (128 every 1 s) to 1 (once every 2 s).
- **For requesting unicast Delay_Resp messages:** The suggested value of logInterMessagePeriod is –4 (16 every 1 s). The suggested configurable range is –7 (128 every 1 s) to 6 (once every 64 s).

The suggested value of durationField in each REQUEST_UNICAST_TRANSMISSION TLV is 300 (300 s) and the suggested configurable range is 10 to 1000.

The maintenance and configuration of these default and configuration range values is implementation specific.

In implementing the GRANT_UNICAST_TRANSMISSION TLV mechanism, the granted values preferably are the same as requested in the received REQUEST_UNICAST_TRANSMISSION TLV as long as the requests are in the configurable range.

NOTE—Since the transport can be unreliable, it is recommended that the requesting PTP Port repeat the request after an implementation-specific timeout if no grant TLV has been received. For receiving continuous service, a requester can reissue a request in advance of the end of the grant period. The recommended advance includes sufficient margin for reissuing the request at least two more times if no grant is received.

The values of defaultDS.announceReceiptTimeout, defaultDS.priority1, defaultDS.priority2, defaultDS.slaveOnly, and τ are identical to those specified in I.3 and I.4.

The physical requirements are identical to those specified in I.3 and I.4.

IECNORM.COM : Click to view the full PDF of IEC 61588:2021

Annex B

(informative)

Timescales and epochs in PTP

B.1 General considerations

A more detailed discussion of many of the topics in this annex can be found in Allan et al. [B1], Allan et al. [B2], IETF RFC 5905 (2010) [B19], ISO 8601:2004 [B29], the USNO Time Service Department website,²³ and Sullivan et al. [B52].

Within a domain, the characteristics of the time are determined by the Grandmaster PTP Instance of the domain. The Grandmaster Clock determines:

- a) The rate at which time advances. The frequency accuracy of the Grandmaster Clock is measured by how well a time interval determined between any two events, as measured using the Grandmaster Clock, corresponds to the same interval measured using a PTP Instance, consistent with the internationally defined second or the time unit in use for the ARB timescale. The internationally defined SI second is the measure of time defining the TAI timescale maintained by the Bureau International des Poids et Mesures near Paris.
- b) The origin, or epoch, of the timescale.

The possible timescales and epochs available for use by the Grandmaster PTP Instance are as follows:

- **timescale PTP:** Indicated by a `timePropertiesDS.ptpTimescale` value of TRUE. The epoch is the PTP epoch.
- **timescale ARB:** Indicated by a `timePropertiesDS.ptpTimescale` value of FALSE. The epoch is specific to the implementation.

B.2 UTC, TAI and the PTP epoch and timescale updates

B.2.1 General properties of UTC, TAI, and the PTP epoch

See NIST SP 330:2008 [B42], with the further amplification of *Proceedings of the 21st General Assembly of the IAU* [B48], for the definition of TAI, and Petit and Luzum [B47] for more information on TAI.

TAI and UTC are international standards for time based on the SI second (see NIST SP 330:2008 [B42], *Proceedings of the 21st General Assembly of the IAU* [B48], and Petit and Luzum [B47]). The SI second is the duration of 9 192 631 770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium 133 atom (see NIST SP 330:2008 [B42], *Proceedings of the 21st General Assembly of the IAU* [B48], and Petit and Luzum [B47] for more details on UTC, TAI, and the SI second). TAI stability is established from a weighted average of clocks in timing laboratories throughout the world. The rate is steered to laboratory frequency standards that best realize this definition, adjusted to the rate on Earth's geoid. $UTC = TAI - \langle dLS \rangle$ (see 7.2.4). The difference between TAI and UTC is defined in IERS Bulletin C. The history of this difference is maintained by the United States Naval Observatory [B54].

²³ USNO Time Service Department.(<http://tycho.usno.navy.mil/>).

UTC is the basis for civil time. The UTC printed representation is specified by ISO 8601:2004 [B29] as YYYY-MM-DD for the date and hh:mm:ss for the time in each day. The rate at which UTC time advances is identical to the rate at which TAI time advances. The UTC time differs from the TAI time by a constant offset. This offset is modified on occasion by adding or subtracting leap seconds.

Starting on 0 h on 1 January 1972 UTC [Modified Julian Day (MJD) 41 317.0], the world's standard time systems began the implementation of leap seconds to allow only integral second correction between UTC and TAI, both of which are expressed in years, months, days, hours, minutes, and seconds. On this date, TAI – UTC was 10 s. Prior to 1 January 1972, corrections to the offset between UTC and TAI were made in fractions of a second; in addition, the duration of the UTC second was not exactly equal to the duration of the TAI second and was different during different periods (see IERS [B49]).

Leap-second corrections, which are applied to UTC but not to TAI, are made preferably following second 23:59:59 of the last day of June or December. The first such correction, a single, positive leap-second correction, was made following 23:59:59 on 30 June 1972 UTC, and UTC was 11 s behind TAI following that instant.

NOTE—As of 0 h on 1 January 2017 UTC, TAI – UTC = +37 s.

In computer networks, the common Portable Operating System Interface (POSIX)-based time conversion algorithms are typically used to produce the correct ISO 8601:2004 [B29] printed representations for both TAI and UTC.

The PTP epoch is set such that a direct application of the POSIX algorithm to a timestamp in the timescale PTP converts this PTP timestamp to the ISO 8601:2004 [B29] printed representation of TAI. PTP also distributes the current offset between TAI and UTC, that is, <dLS>, in the currentUtcOffset field of Announce messages. Except during leap seconds, subtracting <dLS> from a PTP timestamp and then applying the POSIX algorithm results in the ISO 8601:2004 printed representation of UTC. Conversely, except during leap seconds, applying the inverse POSIX algorithm and adding <dLS> converts from the ISO 8601:2004 printed form of UTC to the form required to generate a PTP timestamp.

For example, at 0 h 2 January 1972 TAI, the value of PTP Instance Time was 63 158 400. At this time, <dLS> was 10. The POSIX algorithm applied to the value (63,158,400 – 10) gives a value of 23:59:50 1972-01-01 (10 s before 0 h 2 January 1972 UTC). The value of PTP Instance Time on 0 h 2 January 1972 TAI is computed by observing that PTP Instance Time = 0 on 0 h 1 January 1970 TAI, that is, MJD 40 587. On 0 h 2 January 1972 TAI, MJD = 41 318. Thus, PTP Instance Time on 0 h 2 January 1972 TAI is $0 + 86\,400 \times (41\,318 - 40\,587)$. Note that if this calculation were done for a day in which a leap second occurred, a more complex algorithm would be required to ensure that, for the duration of the leap second, the ISO 8601:2004 [B29] print form seconds value would be 60 for a positive leap second.

International standards specify that if a correction to UTC relative to TAI is required, the leap second occurs at the last second of the UTC day, preferably at the end of June 30 or December 31. For a negative leap second, the last minute of the designated day has only 59 seconds. Negative leap seconds have never occurred and are unlikely to occur in the future. For a positive leap second, the last minute of the designated day has 61 seconds.

Although a negative leap second is unlikely to occur, if such a correction becomes necessary the ISO 8601:2004 [B29] printed representation would appear as follows for a hypothetical negative leap second on 30 June 1972 UTC:

1972-06-30 23:59:57, 1972-06-30 23:59:58, 1972-07-01 00:00:00

For the positive leap second that actually occurred on 30 June 1972 UTC, the ISO 8601:2004 [B29] printed representation appeared as follows:

1972-06-30 23:59:59, 1972-06-30 23:59:60, 1972-07-01 00:00:00

Note the 23:59:60 notation to indicate the added second.

The update semantics for leap-second updates in PTP are discussed in B.2.2.

B.2.2 Update semantics for UTC and alternate timescales

B.2.2.1 General Semantics of timePropertiesDS.currentUtcOffset, timePropertiesDS.leap59, and timePropertiesDS.leap61

When the timescale is PTP, the members of the timePropertiesDS enable PTP Instances of a PTP Network to compute the current UTC time from the PTP Instance Time. PTP distributes the current time on the timescale via the timestamps in the PTP timing messages. PTP does not distribute UTC but distributes the information needed to compute UTC from PTP Instance Time via the timePropertiesDS information contained in Announce messages.

In all cases, per the specifications of 9.4, the PTP Grandmaster Instance is responsible for inserting the correct values of leap-second information into the PTP messages. The PTP Grandmaster Instance is also responsible for receiving this information from an external source.

PTP Instances other than the Grandmaster PTP Instance propagate this information based on the specifications of 9.3.5. Such PTP Instances are able to compute the precise PTP Instance Time for update and the updated value of <dLS> based on the information in their timePropertiesDS without waiting for receipt of the updated <dLS> information from the Grandmaster PTP Instance. Specifically, a PTP Instance can use the values of timePropertiesDS.leap59 and timePropertiesDS.leap61 to compute the PTP Instance Time when the update is to occur. The updated value of <dLS> can be computed from timePropertiesDS.currentUtcOffset and knowing, for example, that for a TRUE value of timePropertiesDS.leap61 that the updated <dLS> will be timePropertiesDS.currentUtcOffset + 1. After a leap-second event, the fields related to leap seconds in the received Announce message will not be updated immediately due to the delay in the propagation of Announce messages. As a result, PTP Instances need to avoid further leap-second processing until updated timePropertiesDS information is received from the Grandmaster PTP.

It is also possible for a receiving PTP Instance to ignore the leap-second flags and simply update the computed values of UTC whenever an updated value of timePropertiesDS.currentUtcOffset is received. However, this option ensures that the updates will not occur at the same instant in all PTP Instances and in all cases will differ from the Grandmaster PTP Instance during the time it takes for the information to propagate according to the specifications of 9.3.5.

B.2.2.2 Semantics for the use of option 16.3 to compute Pacific Standard Time during a positive leap second

Consider the operation of a Grandmaster PTP Instance, and another PTP Instance, C, during the time period from 1990-Dec-31 23:59:59 to 1991-Jan-1 00:00:00 UTC. This time interval includes a positive leap second as the last second of 1990-Dec-31 UTC, which results in the value of <dLS> changing from 25 s to 26 s at midnight of that day. In this example, the Grandmaster PTP Instance is using the ALTERNATE_TIME_OFFSET_INDICATOR TLV to distribute Pacific Standard Time, PST. Note that PST is 8 h (28 800 s) behind UTC, which in turn is <dLS> behind PTP. The total offset of PST from PTP;

that is, `currentOffset` (see 16.3.3.5), is therefore $\text{--dLS} - 28\,800\text{ s}$. Note that the receiving PTP Instance does not itself correct for the leap second in the received value of `timeOfNextJump`. It is the responsibility of the Grandmaster PTP Instance to correct for `<dLS>` in computing values of `jumpSeconds` and `timeOfNextJump`.

B.3 Standard time sources

There are two standard time sources of particular interest in implementing PTP systems for which UTC traceable time is required by the application.

The first time source is the set of systems implementing the Network Time Protocol (NTP) protocol [B19], widely used in synchronizing computer systems within a campus and around the world. A set of NTP servers, to which NTP clients synchronize, is maintained. These servers themselves are synchronized to timeservers traceable to international standards. UTC time accuracy from NTP systems is usually in the millisecond range. NTP provides the current time and warning flags marking the introduction of a leap-second correction, which is inserted at the end of the current UTC day. NTP does not correct the number of NTP seconds since the NTP epoch whenever a leap-second correction is made. (In other words, the NTP clock effectively stops during a leap second, and the time interval occupied by a leap second is effectively “forgotten” once it has been inserted.) The NTP epoch is 0 h on 1 January 1900. NTP was set at 0 h on 1 January 1972 to 2 272 060 800.0, to agree with UTC. Currently, NTP represents seconds as a 32-bit unsigned integer. NTP therefore rolls over every $2^{32}\text{ s} \approx 136\text{ years}$, with the first such rollover occurring in approximately the year 2036.

The second system of interest is Global Navigation Satellite System (GNSS). For example, the GNSS maintained by the U.S. Department of Defense is the GPS. UTC time accuracy from the GPS system is usually in the 10 ns to 100 ns range. GPS system transmissions represent the time as {GPS Weeks, GPS SecondsInLastWeek}, that is, the number of weeks since the GPS epoch and the number of seconds since the beginning of the current week. From this, GPS Seconds, that is, the number of seconds since the GPS epoch, can be computed. GPS provides the current time, a leap-seconds offset, and warning flags marking the introduction of a leap-second correction. From GPS time, UTC and TAI times can be computed using the information contained in the GPS transmissions. The GPS epoch began at 0 h on 6 January 1980 UTC (MJD 44 244). GPS weeks are represented in the satellite transmissions modulo 1024 weeks $\cong 19.7\text{ years}$. The first such rollover occurred between the weeks of 15 August and 22 August 1999. Many, but not all, commercial systems are believed to have correctly managed this rollover.

Either of these systems is appropriate for use in providing time to a clockClass 6 clock. Relationships between the timescales discussed and examples of times in each system for interesting instants are given in Table B.1. In Table B.1, PTP Seconds refers to the seconds portion of the time distributed by the timescale PTP and, as noted, is referenced to 1 January 1970 TAI.

Table B.1—Relationships between timescales

From	To	Formula
NTP Seconds	PTP Seconds	$\text{PTP Seconds} = \text{NTP Seconds} - 2\,208\,988\,800 + \text{currentUtcOffset}$
PTP Seconds	NTP Seconds	$\text{NTP Seconds} = \text{PTP Seconds} + 2\,208\,988\,800 - \text{currentUtcOffset}$
$\text{GPS Seconds} = (\text{GPS Weeks} \times 7 \times 86\,400) + \text{GPSSecondsInLastWeek}$ (GPS week number needs to include $1024 \times \text{number of rollovers}$)	PTP Seconds	$\text{PTP Seconds} = \text{GPS Seconds} + 315\,964\,819$
PTP Seconds	GPS Seconds	$\text{GPS Seconds} = \text{PTP Seconds} - 315\,964\,819$

B.4 Meaning and uses of the attributes of the timePropertiesDS data set

The Grandmaster PTP Instance determines the time and frequency that are distributed within a PTP domain. The Grandmaster PTP Instance also sets the values of attributes in the timePropertiesDS data set that describe properties of the time and frequency. These attributes are also distributed within a domain.

The attribute timePropertiesDS.timeSource (see 8.2.4.9) indicates the nature of the source of time and frequency distributed by the Grandmaster PTP Instance. As noted in 8.2.4.9 and 7.6.2.8, the source can be internal or external to the Grandmaster PTP Instance.

The attribute timePropertiesDS.ptpTimescale (see 8.2.4.8) indicates whether the timescale of the domain as established by the Grandmaster PTP Instance is the timescale PTP or the timescale ARB,(see 7.2). If the value is PTP, then the epoch or origin of the timescale is the PTP epoch (see 7.2.3).

The attribute timePropertiesDS.frequencyTraceable (see 8.2.4.7) indicates whether the frequency is traceable to international standards. A clock consists of an oscillator and a counter. This attribute refers to the frequency of the oscillator. This effectively defines the realization of the second in the domain. For example, the GPS system distributes a signal traceable to international standards. If the Grandmaster PTP Instance has access to this signal from a GPS receiver and uses this to synchronize the Grandmaster PTP Instance within the accuracy defined in the applicable profile for traceability, then this attribute is TRUE. On the other hand, if the Grandmaster PTP Instance either does not have access to the GPS signal or does not make use of it in synchronizing its Local PTP Clock, then whether or not this attribute is TRUE depends on whether the accuracy of the second realized by the free running local clock agrees with the second defined by international standards to within the traceability accuracy specification of the applicable profile.

It is also possible that the Grandmaster PTP Instance implicitly sets the frequency based on periodic updates of the time. If in this case it can be demonstrated that the frequency derived from the Grandmaster Clock is within the traceability accuracy requirement for frequency established by the applicable profile, then the frequency can be considered traceable. Note that if the ptpTimescale is ARB, indicating that the epoch is arbitrary or not known to be traceable to international standards, that is, the time is not traceable, it is still possible that the frequency defined by the Grandmaster Clock is traceable. Indeed, this is precisely the situation dealt with in ITU-T Recommendation G.8265.1 (July 2014) [B35], which specifies how this standard is used to distribute frequency but not time in a telecommunications application.

The attribute timePropertiesDS.timeTraceable (see 8.2.4.6) indicates whether the time indicated by the Grandmaster Clock is traceable to international standards. For example, the GPS system distributes a signal and data from which the time, traceable to international standards, can be derived. If the Grandmaster PTP Instance has access to this signal and data from a GPS receiver and uses this to periodically update the time of Grandmaster Clock within the accuracy defined in the applicable profile for traceability, then this attribute would be TRUE. On the other hand, if the Grandmaster PTP Instance either does not have access to the GPS signal or does not make use of it, this attribute would normally be FALSE unless it could be otherwise demonstrated that the time was indeed traceable. Similar arguments would apply to other potential sources of time traceable to international standards.

The value of the attribute timePropertiesDS.currentUtcOffset (see 8.2.4.2) indicates the difference between the TAI and UTC timescales (see 7.2.4). This difference is defined by international agreement and is independent of the operation of the PTP protocol. Whether the actual value of the attribute timePropertiesDS.currentUtcOffset is valid is indicated by the value of the attribute timePropertiesDS.currentUtcOffsetValid (see 8.2.4.3). If currentUtcOffset is valid and if the timescale of the domain is PTP, then UTC can be computed from the timescale PTP as outlined in 7.2.4. Even if the value of the currentUtcOffset is valid, if the timescale of the domain is ARB, in general UTC cannot be computed from the timescale of the domain. In this case, the ALTERNATE_TIME_OFFSET_INDICATOR TLV of 16.3 can be used to compute UTC. Most Grandmaster PTP Instances will obtain values for UTC offset from their external primary reference. For example, the GPS system distributes this information.

The attribute `timePropertiesDS.leap59` (see 8.2.4.4) or `timePropertiesDS.leap61` (see 8.2.4.5), when TRUE, indicates that a leap-second correction is to occur at the end of the UTC day. This notification information is included in GPS transmissions and from other sources.

The `timePropertiesDS` attributes `timeSource`, `ptpTimescale`, `frequencyTraceable`, and `timeTraceable` are established by the Grandmaster PTP Instance and distributed to other PTP Instances within the domain. None are used in the operation of the BMCA (see 9.3.2). The Grandmaster PTP Instance can make use of the properties described by these attributes in the determination of `clockClass` (see 7.6.2.5). Applications served by PTP Instances can make use of these attributes in determining how to interpret the time or frequency distributed within the domain; for example, whether the timescale is PTP or ARB is usually critical for applications.

Many applications require determining UTC time from time distributed by PTP. This determination is the responsibility of the application or, if specified in the applicable profile, of PTP Instances delivering time directly to an application. The PTP protocol does not distribute UTC directly but, when possible, does distribute the information needed to compute UTC.

It is possible to compute UTC from time distributed by PTP if all the following conditions hold:

- a) The timescale is PTP as indicated by the `ptpTimescale` attribute.
- b) The `currentUtcOffsetValid` attribute is TRUE, indicating that the `currentUtcOffset` value is valid and can be used in the computations.
- c) The `timeTraceable` attribute is TRUE, indicating that the timescale PTP is traceable to international standards to the accuracy specified in the applicable profile.

NOTE—It is possible but not verifiable based on PTP attributes to compute UTC with lower accuracy than specified by the `timeTraceable` attribute.

If these conditions are met, there are two ways provided in the standard for the computation of UTC from time distributed by PTP within the domain.

The first is a direct computation where $UTC = TAI - \text{currentUtcOffset}$ (see 7.2.4). In effect, subtracting the `currentUtcOffset` value from the time of the Local PTP Clock and then applying the result to the POSIX algorithm yields the correct print form of UTC, except during a positive leap second, when the algorithm needs to be modified to create a seconds value of 60.

The second method is the use of the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV of 16.3. This method requires the Grandmaster PTP Instance to distribute the information needed to compute UTC using this TLV. The computation itself is done in PTP Instances distributing time directly to the application. In this case, the values of `currentUtcOffset`, `leap59`, and `leap61` of the common header are not used by PTP Instances in the computation as in the first method, although they might be useful as part of the computations done by the Grandmaster PTP Instance in populating the fields of the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV.

Note that while the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV of 16.3 can be used to distribute information needed to compute UTC from the timescale PTP, its primary purpose is to compute other timescales. When used for its primary purpose, the Grandmaster PTP Instance uses the TLV to distribute the information to compute the desired alternate timescale based on offsets from the timescale distributed by the PTP Grandmaster Instance, that is, PTP or ARB. For example, the TLV can be used for computing time since some external event known to the Grandmaster PTP Instance, for example, the launch of a rocket, from the timescale in use within the domain.

When the timescale is PTP, time is continuous in the domain. However, if the timescale is ARB, time can be piecewise continuous. There are two ways PTP Instances can manage steps in the UTC offset or similar discontinuities in timescale offsets distributed via the `ALTERNATE_TIME_OFFSET_INDICATOR` TLV.

In the case of UTC and a timescale PTP, a PTP Instance receives time and the currentUtcOffset values from the Grandmaster PTP Instance. The PTP Instance synchronizes its Local PTP Clock to that of the Grandmaster Clock by accounting for the propagation delays using either the peer-to-peer or end-to-end methods. Thus, PTP time at PTP Instances always matches the time of the Grandmaster Clock to within the synchronization accuracy. At a leap-second event, the value of currentUtcOffset changes by a second. The notification of this change takes time to propagate to downstream PTP Instances. Hence, while the downstream PTP Instances could simply compute UTC by subtracting its current known value of currentUtcOffset, the result would be in error during the time taken for the updated value of currentUtcOffset to propagate to the PTP Instances.

Alternatively, the leap59 and leap61 flags of the common header can be used by PTP Instances to ensure that leap-second events occur simultaneously throughout the domain to within the accuracy of synchronization. These flags, when TRUE, indicate that a leap second is to be added or subtracted at the end of the current UTC day. These flags are set sufficiently in advance of the event to permit PTP Instances to make the leap-second correction at the correct UTC time based on the time of the PTP Instance's Local PTP Clock.

In the case where the ALTERNATE_TIME_OFFSET_INDICATOR TLV is used, jumps in the timescale are indicated by the jumpSeconds and timeOfNextJump fields. The values of these fields can be used by a PTP Instance to make offset corrections at the appropriate time on the timescale in use based on the time of the PTP Instance's clock.

Annex C

(normative)

Transport of PTP over User Datagram Protocol over Internet Protocol Version 4

C.1 General

This annex specifies those portions of the PTP standard that are specific to implementations that transport messages over the User Datagram Protocol (UDP), as defined in IETF RFC 768 (1980), and Internet Protocol version 4 (IPv4), as defined in IETF RFC 791 (1981). The specifications in this annex shall apply to all PTP implementations using UDP/IPv4 as a communication service.

The first octet of the PTP message shall immediately follow the final octet of the UDP header.

The transmitting or intermediate PTP Instance may set the UDP checksum to 0.

When using this transport with unicast transmission, modifications to PTP event packets by Transparent Clocks might corrupt applications that incorrectly use the UDP destination port.

NOTE—The UDP destination ports in this annex are values assigned to PTP, and no interference is expected to occur. However, it is known that some applications in use disregard these assignments. It is these applications that are vulnerable to the action of Transparent Clocks.

C.2 UDP port numbers

The UDP destination port of a PTP event message shall be 319²⁴.

The UDP destination port of a multicast PTP general message shall be 320.

The UDP destination port of a unicast PTP general message that is addressed to a PTP Instance shall be 320.

The UDP destination port of a unicast PTP general message that is addressed to a manager shall be the UDP source port value of the PTP message to which this is a response.

C.3 IPv4 multicast addresses

PTP messages shall use the multicast message specified in Table C.1.

²⁴ The Internet Assigned Numbers Authority (IANA) assigned the dedicated port numbers shown to PTP (see <http://www.iana.org/assignments/port-numbers/>).

Table C.1—IPv4 multicast addresses

IANA assigned name ²⁵	PTP Message types	Address
PTP-primary	All except peer-to-peer delay mechanism messages	224.0.1.129
PTP-pdelay	Peer-to-peer delay mechanism messages	224.0.0.107

For PTP messages sent to the PTP-pdelay address, the Time to Live (TTL) field shall be set to 1.

C.4 sdold field values

C.4.1 General

Some Version 2.0 implementations are based on hardware designed for Version 1, “V.1 hardware”. This hardware-assisted timestamping, checked the length of the incoming packet before qualifying the timestamp, and required the UDP payload of the PTP event messages to be at least 124 octets in length. In version 1 hardware the PTP event messages were specified by the controlField (see 13.3.2.13).

This edition of the standard deprecates implementations based on the V.1 hardware. V.1 hardware implementations shall be limited to version 2.0 PTP Nodes. Subclause C.4.2 defines an option, the “version 1 hardware option”, that may be used by implementations that must be backward compatible with PTP Networks containing PTP Nodes or PTP Instances based on V.1 hardware.

The version 1 hardware option shall not be used with any transport except IPv4, that is, Annex C of this standard.

The support for this option should be included in product specifications.

C.4.2 Version 1 hardware option

Operation of this option depends on whether the device is a requestor (see C.4.2.1) or a responder (see C.4.2.2) of version 1 hardware support.

C.4.2.1 Version 1 hardware option—Requestor

Requestors are version 2.0 PTP Instances only. PTP Instances using V.1 hardware request other PTP Instances to transmit padded event messages to the requesting PTP Instance by setting the value of sdoId to 100₁₆ in all Announce and PTP event messages transmitted from the requesting PTP Instance. Requestors support the controlField (see 13.3.2.13).

NOTE—Implementations based on Annex D of IEEE Std 1588-2008 will interpret the majorSdoId portion of the sdoId as the transportSpecific field of that edition; that is, a majorSdoId value of 000₁₆ and 100₁₆ will be interpreted as a transportSpecific field value with bit 0 set to 0 or 1, respectively.

C.4.2.2 Version 1 hardware option—Responder

Responders are PTP instances conformant to this standard and support the version 1 hardware option.

²⁵ The IANA assigned the dedicated multicast addresses along with the IANA names to PTP. These names appear in the IANA listings identifying multicast addresses and names.

If the responder receives a version 2.0 PTP Announce or event message with the value of `sdoId` = 100₁₆, it shall extend the UDP payload of all PTP event messages transmitted to the requesting PTP Instance by adding padding to the end of the PTP event message such that the UDP payload length is equal to or greater than 124 octets. The padding octets shall have all bits zero. Such responders shall also support the `controlField` (see 13.3.2.13).

Padding and `controlField` support once active shall continue until such time as the transmitting PTP Port enters the INITIALIZING state.

Responders shall transmit all PTP messages as specified in 7.1.4.

Responders shall disregard the padding octets of a received PTP message.

C.5 Optional values

For PTP event messages, the value of the differentiated service field in the type of service field should be set to the highest traffic class selector codepoint available.

NOTE—When the layer 2 transport mechanism allows for multiple priorities, it is recommended that the highest priority be used for PTP event messages.

C.6 IPv4 Options

IPv4 options shall not be used.

C.7 Protocol addresses

For any quantity of data type `PortAddress` (see 5.3.6), when the `networkProtocol` member value is UDP/IPv4 (see 7.4.1), the `addressLength` member value shall be 4.

The `addressField` member value shall be the IPv4 address of the port represented as four groups of two hexadecimal digits. For example, the IPv4 address 203.0.113.235 expressed in the usual text notation appears as the octet array CB0071EB₁₆.

Annex D

(normative)

Transport of PTP over User Datagram Protocol over Internet Protocol Version 6

D.1 General

This annex specifies those portions of the PTP standard that are specific to implementations that transport messages over the User Datagram Protocol (UDP), as defined in IETF RFC 768 (1980), and Internet Protocol version 6 (IPv6), as defined in IETF RFC 8200 (2017). The specifications in this annex shall apply to all PTP implementations using UDP/IPv6 as a communication service.

The first octet of the PTP message shall immediately follow the final octet of the UDP header.

A transmitting PTP Instance shall extend the UDP payload of all PTP messages by two octets beyond the end of the PTP message. The contents of the UDP checksum field or the final two octets of the UDP payload may be modified by the initiator or an intermediate PTP Instance to ensure that the UDP checksum remains uncompromised after any modification of PTP fields. This modification to update the UDP checksum may be implemented using the mechanism defined in IETF RFC 1624 (1994). Other than for purposes of calculating the UDP checksum, the contents of the UDP field beyond the end of the PTP fields shall be ignored by the receiver.

D.2 UDP port numbers

The UDP destination port value of an PTP event message shall be 319²⁶.

The UDP destination port value of a multicast PTP general message shall be 320.

The UDP destination port value of a unicast PTP general message that is addressed to a PTP Instance shall be 320.

The UDP destination port value of a unicast PTP general message that is addressed to a manager shall be the UDP source port value of the PTP message to which this is a response.

²⁶ The Internet Assigned Numbers Authority (IANA) assigned the dedicated port numbers shown to PTP (see <http://www.iana.org/assignments/port-numbers/>).

D.3 IPv6 multicast addresses

PTP messages shall use the multicast addresses in Table D.1.

Table D.1—IPv6 multicast addresses

IANA assigned name	PTP Message types	Address (hex)
PTP-primary	All except peer-to-peer delay mechanism messages	FF0X:0:0:0:0:0:181; see NOTE
PTP-pdelay	Peer-to-peer delay mechanism messages	FF02:0:0:0:0:0:6B

NOTE—The hexadecimal values for “X” in the PTP-primary address are defined in IETF RFC 4291 (2006) [B15]. These are as follows:

- 0 reserved
- 1 Interface-Local scope
- 2 Link-Local scope
- 3 reserved
- 4 Admin-Local scope
- 5 Site-Local scope
- 6 (unassigned)
- 7 (unassigned)
- 8 Organization-Local scope
- 9 (unassigned)
- A (unassigned)
- B (unassigned)
- C (unassigned)
- D (unassigned)
- E Global scope
- F reserved

For PTP messages sent to the PTP-pdelay address, the Hop Limit (HL) field shall be set to 1.

D.4 Optional values

For PTP event messages, the value of the Differentiated Service (DS) field in the Traffic Class (TC) field should be set to the highest traffic class selector codepoint available.

NOTE 1— When the layer 2 transport mechanism allows for multiple priorities, the highest priority preferably is used for PTP event messages.

NOTE 2— The use of IPv6 Extension Headers is outside the scope of this standard.

D.5 Protocol addresses

For any quantity of data type PortAddress (see 5.3.6), when the networkProtocol member value is UDP/IPv6 (see 7.4.1):

- The addressLength member value shall be 16.
- The addressField member value shall be the IPv6 address of the port represented as 16 groups of two hexadecimal digits. For example, the IPv6 address 2001:0DB8:85A3:08D3:1332:8A2E:0270:7225 expressed in the usual text notation per IETF RFC 4291 (2006) [B15] appear as the octet array 20010DB885A308D313328A2E02707225.

Annex E

(normative)

Transport of PTP over IEEE 802.3 transports

E.1 General

This annex specifies those portions of the PTP standard that are specific to implementations that transport messages directly over Ethernet frames as specified in IEEE 802 standards.

The first octet of the PTP message shall occupy the first octet of the client data field.

E.2 Ethertype

Ethertype shall be 88F7₁₆.

E.3 Multicast media access control (MAC) addresses

By default, PTP messages shall use MAC addresses as specified in 0.

Table E.1—Multicast MAC addresses

PTP Message types	Address (hex)
All except peer-to-peer delay mechanism messages	01-1B-19-00-00-00
Peer-to-peer delay mechanism messages	01-80-C2-00-00-0E

The OUI value of 00-1B-19 represents the value assigned to this standard by the IEEE Registration Authority²⁷. The MAC address value of 01-1B-19-00-00-00 represents a multicast address derived from the pool of multicast addresses within that space.

The MAC address of 01-80-C2-00-00-0E represents a multicast address derived from the pool of multicast addresses administered by IEEE Std 802.1Q. It is permissible, however, to use address 01-1B-19-00-00-00 or address 01-80-C2-00-00-0E for all PTP messages if such use is defined in the applicable PTP Profile.

NOTE 1—According to the IEEE 802.1 model, it is always possible to use PTP peer-to-peer delay mechanism messages even on ports blocked by Spanning Tree Protocols.

NOTE 2—Per 8.6.3 of IEEE Std 802.1Q-2014, frames containing 01-80-C2-00-00-0E in their destination address field are not relayed by the bridge. Therefore, this address is selected for the peer-to-peer delay mechanism messages because the scope of this address is limited to an individual LAN, and this is the normal case of the PTP peer-to-peer delay mechanism messages. This address is not assigned exclusively to PTP, but rather it is a shared address.

Per port, peer-to-peer delay measurements shall use the egress PTP Port's MAC Address as the source MAC Address in PTP peer-to-peer delay mechanism messages.

²⁷ IEEE Registration Authority (<https://standards.ieee.org/regauth/>).

E.4 majorSdoId field values

The majorSdoId field (see 13.3.2.1) shall be interpreted as a subtype of the Ethertype (see Table 2 in 7.1.4).

If the device recognizes the subtype, then the PTP message is passed to the PTP Instance. If the device does not recognize the subtype, then the message is treated as any other message with an unrecognized Ethertype.

E.5 Optional values

When the Ethernet transport mechanism allows for multiple traffic classes, the highest priority traffic class should be used for PTP event messages.

NOTE—For Ethernet, IEEE Std 802.1Q-2014 discusses the implementation of traffic classes.

E.6 Protocol addresses

For any quantity of data type PortAddress (see 5.3.6), when the networkProtocol member value is IEEE 802.3 (see 7.4.1):

- The addressLength member value shall be 6.
- The addressField member value shall be the six octet source address field of the Ethernet header.

Annex F

(normative)

Transport of PTP over DeviceNET

F.1 Protocol

This annex specifies those portions of the PTP standard that are specific to DeviceNet implementations. The specifications in this annex shall apply to all PTP implementations using DeviceNet as a communication network. For additional information on DeviceNet, consult the DeviceNet specification provided by ODVA.²⁸

NOTE—DeviceNet is also covered by IEC 62026-3:2008.

F.2 message timestamp point

The message timestamp point (see 7.3.4.1) shall correspond to the trailing edge of the sixth bit of the end of frame field of the first fragmented packet of a PTP event message as shown in Figure F.1.

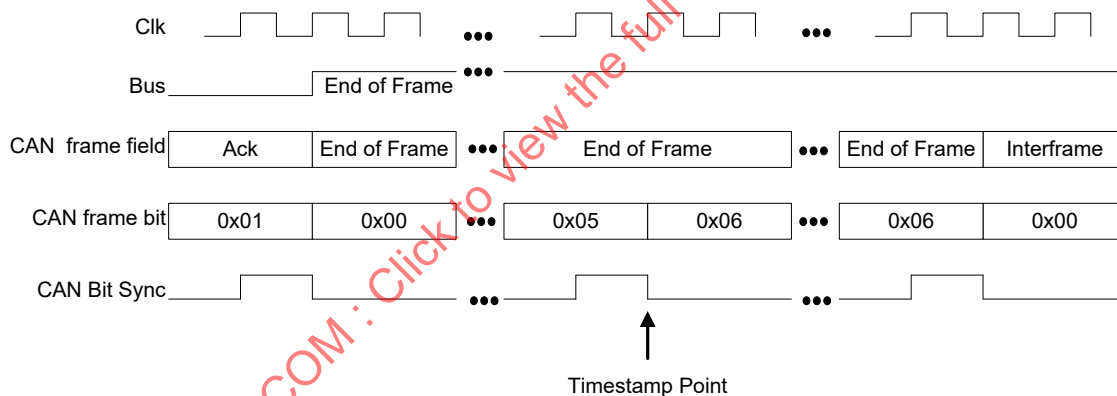


Figure F.1—message timestamp point

F.3 clockIdentity

The clockIdentity for a DeviceNet PTP Node shall be as specified in 7.5.2.2.2.

F.4 PTP message formats

PTP messages are transmitted with the most significant byte of a data type transmitted first followed sequentially by bytes in order of decreasing significance. The first octet of the PTP message shall immediately follow the final octet of the DeviceNet header.

²⁸ Open DeviceNet Vendors Association (<http://www.odva.org/>).

These data are sent using multiple DeviceNet packets (frames) following the standard DeviceNet Explicit Message fragmentation logic. All PTP messages are fragmented on DeviceNet. The DeviceNet header is present in all packets.

DeviceNet headers for all PTP message packets are specified in Table F.1. This header is present in each DeviceNet frame of the PTP message.

Table F.1—DeviceNet headers for all PTP message packets

Octet 0	Octet 1	Octet 2	Type (informative)	Field name
h ₀ h ₁	j ₀ j ₁	k ₀ k ₁	octet octet octet	Fragment = 1, XID = 0, Source MACID Fragment Type, Fragment Count R/R = 1, Service Code = UCMM Service Code

F.5 DeviceNet addressing for PTP

All PTP messages shall be transmitted by an UnConnect Message Manager (UCMM) capable device as an Unconnected Response Message (Message Group 3, Message ID 5) and by a Group 2 Only server as an Unconnected Response Message (Message Group 2, Message ID 3). Thus, each PTP Instance on the subnet has its own unique multicast address [Controller Area Network (CAN) identifier]. The same multicast address is used for all Domains.

All PTP messages shall have the Request/Response bit in the DeviceNet header set to TRUE.

The PTP multicast addresses are shared with other DeviceNet functions, some of which are point-to-point messages. To distinguish a PTP message, the transmitting PTP Instance shall place its own PTP Instance address in the Destination Node field of the DeviceNet message header. The message is then further identified as a PTP message by the UCMM service code.

The UCMM service code field shall be 88 (58₁₆) for the event class of messages and 89 (59₁₆) for the General class of messages.

For any quantity of data type PortAddress (see 5.3.6), when the networkProtocol member value is DeviceNet (see 7.4.1):

- The addressLength member value shall be 2.
- The addressField member value shall be the DeviceNet mac ID.

Annex G

(normative)

Transport of PTP over ControlNet

G.1 Protocol

This annex specifies those portions of the PTP standard that are specific to ControlNet implementations. The specifications in this annex shall apply to all PTP implementations using ControlNet as a communication network. For additional information on ControlNet, consult the ControlNet specification provided by ControlNet International.²⁹

NOTE—ControlNet is also covered by IEC 61158 type 2 elements.

G.2 clockIdentity

The clockIdentity for a ControlNet PTP Node shall be as specified in 7.5.2.2.2.

G.3 PTP message formats

PTP messages are transmitted with the most significant byte of a data type transmitted first followed sequentially by bytes in order of decreasing significance. The first octet of the PTP message shall immediately follow the final octet of the ControlNet LPacket header.

G.4 ControlNet addressing for PTP

The Destination address field for a PTP LPacket shall be 255(FF₁₆) (Broadcast).

The Fixed Tag field for a PTP LPacket shall be 141 (8D₁₆) for event messages and 142(8E₁₆) for general messages.

For any quantity of data type PortAddress (see 5.3.6), when the networkProtocol member value is ControlNet (see 7.4.1):

- The addressLength member value shall be 2.
- The addressField member value shall be the ControlNet PTP Instance number of the device.

²⁹ ControlNet (<http://www.controlnet.org/>).

Annex H

(normative)

Transport of PTP over IEC 61158 Type 10

H.1 Background

PROFINET (IEC 61158 Type 10) specifies a fieldbus communication system. More specific information on how this fieldbus communication system is used to interoperate in a system is given in the communication profiles IEC 61784-1:2007 and IEC 61784-2:2007.

IEC 61784-1:2007 and IEC 61784-2:2007 specify Communication Profile Families (CPFs) and, within a CPF, one or more Communication Profiles (CPs). A CP refers to IEC 61158 Types. IEC 61784-1:2007 specifies various fieldbuses. IEC 61784-2:2007 specifies various real-time Ethernet fieldbuses. PROFIBUS³⁰ and PROFINET are specified in CPF 3. CP 3/4, CP 3/5, and CP 3/6 specify PROFINET in IEC 61784-2:2007.

The IEC 61158 Type 10 protocol is specified in IEC 61158-6-10:2007. IEC 61158 Type 10 services are specified in IEC 61158-5-10:2007.

This annex specifies the protocol used for the transport of PTP over Layer 2 for the CP 3/4, CP 3/5, and CP 3/6 of IEC 61784-2:2007, also known as PROFINET. These CPs refer to IEC 61158-5-10:2007, IEC 61158-6-10:2007, and other standards.

Figure H.1 illustrates a PTP region and an IEC 61158 Type 10 region. A Boundary Clock is used to translate between the protocol in the two regions.

The protocol of this annex is functionally equivalent to Transparent Clock and Ordinary Clock functionality of PTP over Layer 2 in the main subclauses and annexes of this standard. However, the protocol of this annex has a different encoding of the PTP messages to meet the encoding specifications for CP 3/4, CP 3/5, and CP 3/6 of IEC 61784-2:2007 within IEC 61158. This annex is not applicable to CP 3/1, CP 3/2, and CP 3/3 of IEC 61784-1:2007.

NOTE—Existing ASICs support the PTP over Layer 2 of CP 3/4, CP 3/5, and CP 3/6 of IEC 61784-2:2007.

The encoding of this annex shall be used for implementations required to meet the encoding specifications for CP 3/4, CP 3/5, and CP 3/6 of IEC 61784-2:2007 within IEC 61158.

³⁰PROFIBUSTM is the trade name of the nonprofit organization PROFIBUS Nutzerorganisation e.V. (PNO). This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products are acceptable if they can be shown to lead to the same results.

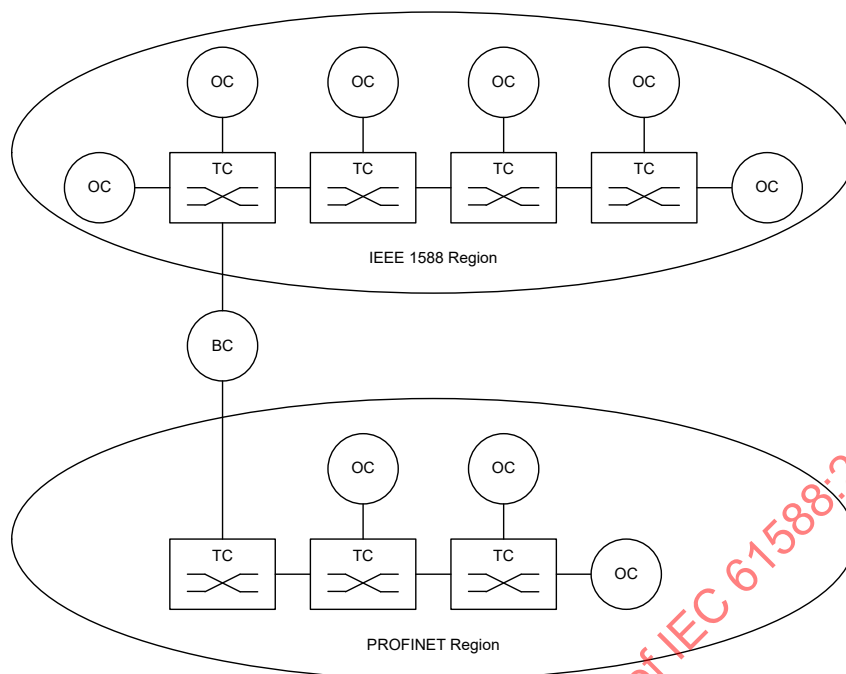


Figure H.1—PROFINET region combined with domains

H.2 Message specification

The mappings of the different message names are provided in Table H.1.

Table H.1—Mapping of messages

Names for PROFINET	Names for PTP
SyncPDU	Sync
FollowUpPDU	Follow_Up
AnnouncePDU	Announce
Not used	Delay_Req
Not used	Delay_Resp
DelayReqPDU	Pdelay_Req
DelayResPDU	Pdelay_Resp
DelayFuResPDU	Pdelay_Resp_Follow_Up
Not used	Signaling
Not used	PTP management

The coding of the PROFINET messages and the used acronyms, abbreviations, and conventions shall be used according IEC 61158-5-10:2007 and IEC 61158-6-10:2007.

For any quantity of data type PortAddress (see 5.3.6), when the networkProtocol member value is PROFINET (see 7.4.1):

- The addressLength member value shall be 6.
- The addressField member value shall be the 6 octet source address of the Ethernet header.

H.3 DLPDU of the IEC 61158 TYPE10

H.3.1 Abstract syntax of the DLPDU

Table H.2 gives an outline of the abstract syntax of the DLPDU according to IEEE Std 802.3.

The encoding and decoding of the fields in Table H.2 shall be according to IEEE Std 802.3 for the DLPDU.

Table H.2—IEEE 802.3 DLPDU syntax

DLPDU name	DLPDU structure
DLPDU	Preamble, ^a StartFrameDelimiter, DestinationAddress, SourceAddress, DLSDU, ^b DLPDU_Padding, ^c FrameCheckSequence
DLSDU	LT, FIDAPDU
FIDAPDU	FrameID, SyncPDU ^ AnnouncePDU ^ FollowUpPDU ^ DelayReqPDU ^ DelayResPDU ^ DelayFuResPDU
NOTE—According to IEEE Std 802.3, the DLPDUs have a minimum length of 64 octets (excluded Preamble, Start Frame Delimiter).	

^a The field contains at least 7 octets.

^b The minimum DLSDU size is 2 octets.

^c The number of padding octets shall be in the range of 0 to 46 depending on the DLSDU size. The value shall be set to zero.

H.3.2 Coding of the DLPDU field DestinationAddress

The DLPDU field shall be coded as data type Octet[6]. The value of the field DestinationAddress shall be an IEEE 802 MAC address.

For PTP over PROFINET-PDUs, the value shall be set according to the coding of the DLPDU DestinationAddress as specified in IEC 61158-6-10, 4.2.2.3.3; Ed3.

H.3.3 Coding of the field LT

The LT field shall be coded with the values according to IEEE Std 802.3 (Unsigned16). This specification uses the values according to Table H.3.

Table H.3—LT (Length/Type)

Value (hex)	Meaning
8892	PROFINET

H.3.4 Coding of the field FrameID

The FrameID field shall be coded as data type Unsigned16 with the values according to Table H.4. This field identifies the structure and the type of the APDU.

Table H.4—FrameID

Value (hex)	Meaning	Use
0000–001F	Reserved	—
0020	SyncPDU	SyncPDU with follow up used for PTP Instance synchronization (isochronous application)
0021	SyncPDU	SyncPDU with follow up used for time synchronization
0022–007F	Reserved	—
0080	SyncPDU	SyncPDU without follow up used for PTP Instance synchronization (isochronous application)
0081	SyncPDU	SyncPDU without follow up used for time synchronization
0082–FEFF	Reserved	—
FF00	AnnouncePDU (clock)	AnnouncePDU is used for synchronization (isochronous application)
FF01	AnnouncePDU (time)	AnnouncePDU is used for time synchronization
FF02–FF1F	Reserved	—
FF20	FollowUpPDU (clock)	FollowUpPDU is used for clock synchronization
FF21	FollowUpPDU (time)	FollowUpPDU is used for time synchronization
FF22–FF3F	Reserved	—
FF40	DelayReqPDU	DelayReqPDU is used for path delay measurement
FF41	DelayResPDU	DelayResPDU is used for path delay measurement with follow up
FF42	DelayFuResPDU	DelayFuResPDU is used for path delay measurement
FF43	DelayResPDU	DelayResPDU is used for path delay measurement without follow up
FF44–FFFF	Reserved	—

H.4 Encoding specifications

A bridge can convert the two formats at the edge. The mapping of the different formats and the different parameter and attribute names are provided in Table H.5. The translation of flagField (see 13.3.2.8) to PROFINET is provided in Table H.6.

Table H.5—Mapping of the parameter and attribute names

Names for PROFINET	Message type	Names for PTP
No counterpart	—	majorSdoId
FrameID	SyncPDU FollowUpPDU AnnouncePDU DelayReqPDU DelayResPDU DelayFuResPDU	messageType
No counterpart	—	versionPTP
No counterpart	—	minorVersionPTP
No counterpart	—	messageLength
SubdomainUUID	SyncPDU FollowUpPDU AnnouncePDU DelayReqPDU DelayResPDU DelayFuResPDU	domainNumber
According to Table 37	SyncPDU	flagField
SequenceId	SyncPDU FollowUpPDU AnnouncePDU DelayReqPDU DelayResPDU DelayFuResPDU	sequenceId
MasterSourceAddress	SyncPDU FollowUpPDU AnnouncePDU	clockIdentity
Is specified in PROFINET	—	logMessageInterval
Seconds	SyncPDU	seconds (Bit 0–31)
NanoSeconds	SyncPDU	Nanoseconds
EpochNumber	SyncPDU	seconds (Bit 32–47)
CurrentUTCOffset	SyncPDU	currentUtcOffset
ClockAccuracy	SyncPDU AnnouncePDU	clockAccuracy
ClockClass	SyncPDU AnnouncePDU	clockClass
MasterPriority1	SyncPDU AnnouncePDU	priority1
MasterPriority2	SyncPDU AnnouncePDU	priority2
ClockVariance	SyncPDU AnnouncePDU	offsetScaledLogVariance
No counterpart	—	stepsRemoved
No counterpart	—	grandmasterIdentity
No counterpart	—	parentPortIdentity
RequestSourceAddress	DelayReqPDU DelayResPDU DelayFuResPDU	clockIdentity
RequestPortID	DelayReqPDU DelayResPDU DelayFuResPDU	portNumber

Table H.6—Translation of flagField from PTP version 2 to PROFINET

Names for PROFINET	Names for PTP version 2.1
Last minute has 61 s	flagField.leap61
Last minute has 59 s	flagField.leap59
Signaled by the AnnouncePDU	flagField.alternateMasterFlag
Coded in FrameID	flagField.twoStepFlag
TRUE for time synchronization and ClockStratum = 1 or 2	flagField.timeTraceable
TRUE for ClockStratum = 1 or 2	flagField.frequencyTraceable
FALSE for clock synchronization (ARP) TRUE for time synchronization if identifier is not INIT or DFLT. Otherwise FALSE	flagField.ptpTimescale
FALSE for clock synchronization (ARP) TRUE for time synchronization if identifier is not INIT or DFLT. Otherwise FALSE	flagField.currentUtcOffsetValid
FALSE	flagField.unicastFlag
Set to FALSE	All other flagField

The coding of the IEC 61158 Type 10 parameter, attributes, and the used acronyms, abbreviations, and conventions shall be used according IEC 61158-5-10:2007 and IEC 61158-6-10:2007.

Annex I

(normative)

Default PTP Profiles

I.1 General

Each default PTP Profile specifies a selection of options and attributes. Each selection specifies a PTP Network that works without requiring user configuration.

I.2 General requirements

PTP Instance shall implement all requirements in the respective PTP Profile that specify default values or choices such that these default values or choices apply without requiring user configuration, that is, as delivered from the manufacturer.

I.3 Delay Request-Response Default PTP Profile

I.3.1 Identification

The identification values for this PTP Profile (see 20.3.3) are as follows:

- PTP Profile
- Default PTP Profile for use with the delay request-response mechanism
- `profileName`: Default delay request-response profile
- `profileNumber`: 1
- `primaryVersion`: 1
- `revisionNumber`: 0
- `profileIdentifier`: 00-1B-19-01-01-00

This profile is specified by the IEEE Precise Networked Clock Synchronization Working Group of the IM/ST Committee.

A copy can be obtained by ordering IEEE Std 1588-2019 from the IEEE Standards Organization <https://standards.ieee.org>.

I.3.2 PTP attribute values

All PTP Instances shall support the ranges and shall have the default initialization values for attributes as follows:

- **defaultDS.domainNumber**: The default initialization value shall be 0.
- **portDS.logAnnounceInterval**: The default initialization value shall be 1. The configurable range shall be 0 to 4.
- **portDS.logSyncInterval**: The default initialization value shall be 0. The configurable range shall be –1 to +1.

- **portDS.logMinDelayReqInterval:** The default initialization value shall be 0. The configurable range shall be 0 to 5.
- **portDS.announceReceiptTimeout:** The default initialization value shall be 3. The configurable range shall be 2 to 10.
- **defaultDS.priority1:** The default initialization value shall be 128.
- **defaultDS.priority2:** The default initialization value shall be 128.
- **defaultDS.slaveOnly:** If this parameter is configurable, the default initialization value shall be FALSE.
- **τ (see 7.6.3.2):** The value shall be 1.0 s.
- **defaultDS.sdoId:** The default initialization value shall be 000₁₆.

For each defined range, manufacturers may support wider ranges within the constraints specified in this standard, for example, restrictions on domainNumber values in Table 2.

I.3.3 PTP Options

All options of Clause 16 and Clause 17 are permitted. By default, these options shall be inactive (disabled) unless specifically activated (enabled) by a management mechanism.

The optional provision of item e) in 9.3.2.5 is permitted. By default, this optional provision shall be inactive (disabled) unless specifically specified-by-design or activated (enabled) by a management mechanism.

See 8.1.4.3 for permitted management mechanism options.

The best master clock algorithm shall be the algorithm specified in 9.3.2.

The delay request-response mechanism shall be the default path delay measurement mechanism. The peer-to-peer delay mechanism may also be implemented.

NOTE—Only a single mechanism is allowed per path, that is, PTP Communication Path or PTP Link. Boundary Clocks are typically used between links that use different path delay mechanisms.

I.3.4 Clock physical requirements

I.3.4.1 Frequency accuracy

Every Grandmaster Clock shall maintain a frequency such that the value of the second as measured by the Grandmaster Clock deviates no more than 0.01% from the second defined by the grandmaster's timescale (see 7.2.1).

I.3.4.2 Frequency adjustment range

Any PTP Instance with a port in the in the SLAVE state shall be able to correct the frequency of its Local PTP Clock to match the frequency of any Master Clock meeting the requirements of I.3.4.1.

NOTE—The frequency adjustment range are typically at least $\pm 0.025\%$.

I.4 Peer-to-Peer Default PTP Profile

I.4.1 Identification

The identification values for this PTP Profile (see 20.3.3) are as follows:

- PTP Profile
- Default PTP Profile for use with the peer-to-peer delay mechanism
- `profileName`: Default delay peer-to-peer delay profile
- `profileNumber`: 2
- `primaryVersion`: 1
- `revisionNumber`: 0
- `profileIdentifier`: 00-1B-19-02-01-00

This profile is specified by the IEEE Precise Networked Clock Synchronization Working Group of the IM/ST Committee.

A copy can be obtained by ordering IEEE Std 1588-2019 from the IEEE Standards Organization <https://standards.ieee.org>.

I.4.2 PTP attribute values

All PTP Instances shall support the ranges and shall have the default initialization values for attributes as follows:

- **defaultDS.domainNumber**: The default initialization value shall be 0.
- **portDS.logAnnounceInterval**: The default initialization value shall be 1. The configurable range shall be 0 to 4.
- **portDS.logSyncInterval**: The default initialization value shall be 0. The configurable range shall be –1 to +1.
- **portDS.logMinPdelayReqInterval**: The default initialization value shall be 0. The configurable range shall be 0 to 5.
- **portDS.announceReceiptTimeout**: The default initialization value shall be 3. The configurable range shall be 2 to 10.
- **defaultDS.priority1**: The default initialization value shall be 128.
- **defaultDS.priority2**: The default initialization value shall be 128.
- **defaultDS.slaveOnly**: If this parameter is configurable the default initialization value shall be FALSE.
- **τ (see 7.6.3.2)**: The value shall be 1.0 s.
- **defaultDS.sdoId**: The default initialization value shall be 000₁₆.
- **transparentClockdefaultDS.primaryDomain**: If implemented, its default initialization value shall be 0.

For each defined range, manufacturers may support wider ranges within the constraints specified in this standard, for example, restrictions on domainNumber values in Table 2.

I.4.3 PTP options

All options of Clause 16 and Clause 17 are permitted. By default, these options shall be inactive (disabled) unless specifically activated (enabled) by a management procedure.

The optional provision of item e) in 9.3.2.5 is permitted. By default, this optional provision shall be inactive (disabled) unless specifically specified-by-design or activated (enabled) by a management mechanism.

See 8.1.4.3 for permitted management mechanism options.

The best master clock algorithm shall be the algorithm specified in 9.3.2.

The peer-to-peer delay mechanism shall be the default path delay measurement mechanism. The delay request-response mechanism may also be implemented.

NOTE—Only a single mechanism is allowed per path, that is, PTP Communication Path or PTP Link. Boundary Clocks are typically used between links that use different path delay mechanisms.

I.4.4 Clock physical requirements

I.4.4.1 Frequency accuracy

Every Grandmaster Clock shall maintain a frequency such that the value of the second as measured by the Grandmaster Clock deviates no more than 0.01% from the second defined by the grandmaster's timescale (see 7.2.1).

I.4.4.2 Frequency adjustment range

Any PTP Instance with a PTP Port in the SLAVE state shall be able to correct the frequency of its Local PTP Clock to match the frequency of any Master Clock meeting the requirements of I.4.4.1.

NOTE—The frequency adjustment range are typically at least $\pm 0.025\%$.

I.5 High-Accuracy Delay Request-Response Default PTP Profile

I.5.1 Identification

The identification values for this PTP Profile (see 20.3.3) are as follows:

- **PTP Profile:** Default PTP Profile for use with the hardware that provides High Accuracy support
- **profileName:** High Accuracy Delay Request-Response Default PTP Profile
- **profileNumber:** 3
- **primaryVersion:** 1
- **revisionNumber:** 0
- **ProfileIdentifier:** 00-1B-19-03-01-00