Document Title	Time Synchronization Protocol Specification
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	897

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Offset Time Bases removed
2023-11-23	R23-11	AUTOSAR Release Management	• Integrated Support of PTP physical clock adjustment and Introduce IEEE 1722 related features handling of streams and tunneling legacy communication (CAN and LIN) • updates of ranges in domainNumber and OfsTimeDomain
2022-11-24	R22-11	AUTOSAR Release Management	• Support for "Secured Time Synchronization" added • Support for time synchronization on peer-to-peer and multidrop topologies added • AUTOSAR TLV processing enhanced • Use case table removed
2021-11-25	R21-11	AUTOSAR Release Management	• Inconsistent handling of Sequence Counter jumps resolved • Trace IDs clean-up
2020-11-30	R20-11	AUTOSAR Release Management	• Added Sequence Counter handling • New configuration parameters

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Clarified SGW value handling for missing Sub-TLVs • Changed Document Status from Final to published
2019-03-29	1.5.1	AUTOSAR Release Management	• Minor changes
2018-10-31	1.5.0	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Contents

1 Introduction and overview

This protocol specification specifies the format, message sequences and semantics of the AUTOSAR Time synchronization Protocol.

The Time synchronization Protocol handles the distribution of time information over Ethernet. The Ethernet mechanism is based on existing PTP (Precision Time Protocol) mechanisms that are described in standards like IEEE1588 (IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems) and IEEE802.1AS (Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks). IEEE802.1AS, also known as gPTP (generalized Precision Time Protocol), can be seen as a profile (or subset) for using IEEE1588. However, neither IEEE1588 nor IEEE802.1AS have been developed considering automotive requirements. Therefore, the Time Synchronization over Ethernet uses the current mechanisms as defined in IEEE802.1AS with specific extensions and/or restrictions. Automotive Ethernet networks deviate from commercial Ethernet networks in terms of the following items:

- Role and functions of ECUs is known and defined a priori
- The network is static, i.e. components like ECUs, switches and characteristics like cable length, don't change during operation or even after switching off and switching on the vehicle. Components of course may be unavailable (due to failure situations or by purpose) but mostly only change when the vehicle is at a service facility.

Therefore, dynamic mechanisms like determining the Global Time Master (denoted as grandmaster in IEEE802.1AS) by the best master clock algorithm (BMCA) during operation are not required. It is also possible to omit the cyclic measurement of link delays on Ethernet links due to the static nature of the automotive network and restrict mechanisms that belonging to dynamic network topology.

1.1 Protocol purpose and objectives

The Time synchronization protocol is used to

- synchronize time bases and the corresponding Ethernet messages
- measure time differences between Ethernet frames

1.2 Applicability of the protocol

The concept is targeted at supporting time-critical and safety-related automotive applications such as airbag systems and braking systems. This doesn't mean that the concept has all that is required by such systems though, but crucial timing-related features that cannot be deferred to implementation are considered.

The concept is also targeted to secure the time bases to support security-critical use-cases such as digital certificate validity check and secure logging. It is also important to secure the time bases used in time-critical and safety-related automotive applications.

1.2.1 Constraints and assumptions

This document specifies the AUTOSAR Time Synchronization Protocol. It was created during elaboration of the AUTOSAR Foundation Standard 1.5.0 which took place in parallel to the development of the AUTOSAR Classic Standard 4.4.0. It already reflects all changes implied to TimeSyncOverEthernet by the work which was done for AUTOSAR Classic Platform.

1.2.2 Limitations

- No support of BMCA protocol, like specified in [1, IEEE 802.1 AS]
- No support of Announce and Signaling messages, like specified in [1, IEEE 802.1 AS].
- The reception of a Pdelay_Req is not taken as a pre-condition to start with the transmission of Sync messages.
- While IEEE 802.1AS states, that IEEE 802.1AS message shall not have a VLAN tag nor a priority tag, the Time synchronization protocol would allow Time Synchronization on VLANs under the condition, that the switch HW supports forwarding of reserved multicast addresses using the range of 01:80:C2:00:00:00 .. 0F
- 'CRC secured' in the context of this document refers to CRC integrity protection mechanism and does not imply that CRC is used as a cybersecurity solution.
- No support of securing the messages of Pdelay protocol.

1.2.3 Accuracy

The accuracy of Time Synchronization depends on various factors (e.g., oscillator accuracy, number of bridges in the network path, configuration, ...). Refer to [2, EXP Time Sensitive Network Features], chapter "Accuracy of Time Synchronization", for recommendations on how to properly configure the overall system for highest possible accuracy.

1.3 Dependencies

1.3.1 Dependencies to other protocol layers

There are no dependencies to other protocols.

1.3.2 Dependencies to other standards and norms

The AUTOSAR Time Synchronization protocol is derived from [1, IEEE 802.1 AS]. For VLAN characteristics refer to [3, IEEE 802.1Q].

1.3.3 Dependencies to the Application Layer

There are no dependencies to the application layer.

2 Protocol Requirements

2.1 Requirements Traceability

Requirement	Description	Satisfied by
[RS_TS_00039]	The implementation of Time Synchronization shall provide Freshness Value (FV) to TSP modules required to secure the time information	[PRS_TS_00249] [PRS_TS_00250]
[RS_TS_20047]	The Timesync over Ethernet module shall trigger Time Base Synchronization transmission	[PRS_TS_00016] [PRS_TS_00050] [PRS_TS_00186] [PRS_TS_00242]
[RS_TS_20048]	The Timesync over Ethernet module shall support IEEE 802.1AS as well as AUTOSAR extensions	[PRS_TS_00002] [PRS_TS_00003] [PRS_TS_00004] [PRS_TS_00005] [PRS_TS_00011] [PRS_TS_00012] [PRS_TS_00016] [PRS_TS_00018] [PRS_TS_00023] [PRS_TS_00025] [PRS_TS_00028] [PRS_TS_00050] [PRS_TS_00053] [PRS_TS_00054] [PRS_TS_00055] [PRS_TS_00056] [PRS_TS_00057] [PRS_TS_00058] [PRS_TS_00059] [PRS_TS_00060] [PRS_TS_00061] [PRS_TS_00062] [PRS_TS_00063] [PRS_TS_00066] [PRS_TS_00067] [PRS_TS_00068] [PRS_TS_00069] [PRS_TS_00070] [PRS_TS_00071] [PRS_TS_00075] [PRS_TS_00077] [PRS_TS_00079] [PRS_TS_00104] [PRS_TS_00141] [PRS_TS_00142] [PRS_TS_00149] [PRS_TS_00154] [PRS_TS_00163] [PRS_TS_00164] [PRS_TS_00166] [PRS_TS_00167] [PRS_TS_00168] [PRS_TS_00169] [PRS_TS_00170] [PRS_TS_00171] [PRS_TS_00181] [PRS_TS_00206] [PRS_TS_00207] [PRS_TS_00208] [PRS_TS_00209] [PRS_TS_00210] [PRS_TS_00219] [PRS_TS_00256] [PRS_TS_00257] [PRS_TS_00262] [PRS_TS_00264] [PRS_TS_00265]
[RS_TS_20051]	The Timesync over Ethernet module shall detect and handle errors in synchronization protocol / communication	[PRS_TS_00004] [PRS_TS_00025] [PRS_TS_00164] [PRS_TS_00210] [PRS_TS_00219]
[RS_TS_20052]	The configuration of the Time Synchronization over Ethernet module shall allow the module to work as a Time Master	[PRS_TS_00094]
[RS_TS_20053]	The configuration of the Time Synchronization over Ethernet module shall allow the module to work as a Time Slave	[PRS_TS_00156]
[RS_TS_20054]	The Implementation of the Time Synchronization shall evaluate and propagate Time Gateway relevant information	[PRS_TS_00094] [PRS_TS_00156] [PRS_TS_00211] [PRS_TS_00212]

▽

△

Requirement	Description	Satisfied by
[RS_TS_20059]	The Timesync over Ethernet module shall access all communication ports belonging to Time Synchronization	[PRS_TS_00053] [PRS_TS_00054] [PRS_TS_00055] [PRS_TS_00056] [PRS_TS_00057] [PRS_TS_00058] [PRS_TS_00059] [PRS_TS_00060] [PRS_TS_00166] [PRS_TS_00167] [PRS_TS_00168] [PRS_TS_00169] [PRS_TS_00170] [PRS_TS_00171] [PRS_TS_00207] [PRS_TS_00208] [PRS_TS_00209]
[RS_TS_20061]	The Timesync over Ethernet module shall support means to protect the Time Synchronization protocol	[PRS_TS_00028] [PRS_TS_00062] [PRS_TS_00063] [PRS_TS_00066] [PRS_TS_00067] [PRS_TS_00068] [PRS_TS_00069] [PRS_TS_00070] [PRS_TS_00071] [PRS_TS_00074] [PRS_TS_00075] [PRS_TS_00076] [PRS_TS_00077] [PRS_TS_00078] [PRS_TS_00079] [PRS_TS_00091] [PRS_TS_00092] [PRS_TS_00093] [PRS_TS_00097] [PRS_TS_00098] [PRS_TS_00099] [PRS_TS_00100] [PRS_TS_00101] [PRS_TS_00102] [PRS_TS_00104] [PRS_TS_00105] [PRS_TS_00106] [PRS_TS_00107] [PRS_TS_00108] [PRS_TS_00109] [PRS_TS_00112] [PRS_TS_00113] [PRS_TS_00114] [PRS_TS_00115] [PRS_TS_00116] [PRS_TS_00118] [PRS_TS_00119] [PRS_TS_00120] [PRS_TS_00157] [PRS_TS_00181] [PRS_TS_00182] [PRS_TS_00183] [PRS_TS_00184] [PRS_TS_00185] [PRS_TS_00187] [PRS_TS_00188] [PRS_TS_00189] [PRS_TS_00190] [PRS_TS_00191] [PRS_TS_00192] [PRS_TS_00193] [PRS_TS_00194] [PRS_TS_00195] [PRS_TS_00196] [PRS_TS_00197] [PRS_TS_00214] [PRS_TS_00215] [PRS_TS_00217] [PRS_TS_00257] [PRS_TS_00266] [PRS_TS_00267] [PRS_TS_00269] [PRS_TS_00270] [PRS_TS_00271] [PRS_TS_00272] [PRS_TS_00273] [PRS_TS_00274] [PRS_TS_00275]
[RS_TS_20062]	The Timesync over Ethernet module shall support user specific data within the time measurement and synchronization protocol	[PRS_TS_00028] [PRS_TS_00062] [PRS_TS_00063] [PRS_TS_00066] [PRS_TS_00067] [PRS_TS_00068] [PRS_TS_00069] [PRS_TS_00070] [PRS_TS_00071] [PRS_TS_00074] [PRS_TS_00075] [PRS_TS_00076] [PRS_TS_00077] [PRS_TS_00078] [PRS_TS_00079] [PRS_TS_00092] [PRS_TS_00104] [PRS_TS_00105] [PRS_TS_00106] [PRS_TS_00118] [PRS_TS_00119] [PRS_TS_00120] [PRS_TS_00181] [PRS_TS_00217] [PRS_TS_00218] [PRS_TS_00256] [PRS_TS_00257]

▽

△

Requirement	Description	Satisfied by
[RS_TS_20066]	The Timesync over Ethernet module shall support measuring the peer-to-peer delay using the IEEE 802.1AS peer-to-peer delay mechanism.	[PRS_TS_00003] [PRS_TS_00011] [PRS_TS_00012] [PRS_TS_00140] [PRS_TS_00141] [PRS_TS_00142] [PRS_TS_00143] [PRS_TS_00149] [PRS_TS_00262] [PRS_TS_00264] [PRS_TS_00265]
[RS_TS_20071]	The Timesync over Ethernet module shall enable time synchronization on peer-to-peer and multidrop topologies	[PRS_TS_00219]
[RS_TS_20072]	The Timesync over Ethernet module shall support means to secure the Time Synchronization protocol	[PRS_TS_00063] [PRS_TS_00071] [PRS_TS_00093] [PRS_TS_00105] [PRS_TS_00107] [PRS_TS_00108] [PRS_TS_00109] [PRS_TS_00220] [PRS_TS_00221] [PRS_TS_00222] [PRS_TS_00223] [PRS_TS_00224] [PRS_TS_00225] [PRS_TS_00226] [PRS_TS_00227] [PRS_TS_00228] [PRS_TS_00229] [PRS_TS_00230] [PRS_TS_00231] [PRS_TS_00232] [PRS_TS_00233] [PRS_TS_00234] [PRS_TS_00235] [PRS_TS_00236] [PRS_TS_00237] [PRS_TS_00238] [PRS_TS_00239] [PRS_TS_00240] [PRS_TS_00241] [PRS_TS_00242] [PRS_TS_00243] [PRS_TS_00244] [PRS_TS_00245] [PRS_TS_00246] [PRS_TS_00247] [PRS_TS_00248] [PRS_TS_00249] [PRS_TS_00250] [PRS_TS_00251] [PRS_TS_00252] [PRS_TS_00253] [PRS_TS_00254] [PRS_TS_00255] [PRS_TS_00257] [PRS_TS_00258]
[RS_TS_20075]	Rate Ratio Calculation	[PRS_TS_00259] [PRS_TS_00260] [PRS_TS_00261] [PRS_TS_00263]

Table 2.1: Requirements Tracing

3 Definition of terms and acronyms

3.1 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
(G)TD	(Global) Time Domain
(G)TM	(Global) Time Master
<Bus>TSyn	A bus specific Time Synchronization module
AVB	Audio Video Bridging
BMCA	Best Master Clock Algorithm
CID	Company ID (IEEE)
CRC	Cyclic Redundancy Checksum
Debounce Time	Minimum gap between two Tx messages with the same PDU
ETH	Ethernet
EthTSyn	Time Synchronization Provider module for Ethernet
Follow_Up	Time transport message (Follow-Up)
GM(C)	Grand Master (Clock)
ICV	Integrity Check Value
IDS	Intrusion Detection System
Pdelay	Propagation / path delay as given in IEEE 802.1AS
Pdelay_Req	Propagation / path delay request message
Pdelay_Resp	Propagation / path delay response message
Pdelay_Resp_Follow_Up	Propagation / path delay Follow-Up message
PDU	Protocol Data Unit
PTP	Precision Time Protocol
StbM	Synchronized Time-Base Manager
Timesync	Time Synchronization
Sync	Time synchronization message (Sync)
TG	Time Gateway
TLV	Type/Tag-Length-Value encoding scheme used by various protocols (e.g. IEEE 802.1AS) to encode data elements
TS	Time Slave
TSD	Time Sub-domain
VLAN	Virtual Local Area Network
linkDelay	neighborPropDelay as defined by [1, IEEE 802.1 AS]
neighborRateRatio	Neighbor Rate Ratio between the local clocks of the Peer Delay Responder and the Peer Delay Initiator according to as defined by [1, IEEE 802.1 AS] (refer to [PRS_TS_00259])
cumulativeScaledRateOffset	cumulativeScaledRateOffset as defined by [1, IEEE 802.1 AS]
t1	Egress timestamp of the `Pdelay_Req` message on Peer Delay Initiator side (refer to Figure 4.1)
t2	Ingress timestamp of the `Pdelay_Req` message on Peer Delay Responder side (refer to Figure 4.1)
t3	Egress timestamp of the `Pdelay_Resp` message on Peer Delay Responder side (refer to Figure 4.1)
t4	Ingress timestamp of the `Pdelay_Resp` message on Peer Delay Initiator side (refer to Figure 4.1) to [PRS_TS_00259]

4 Protocol specification

4.1 General

[PRS_TS_00002]

Upstream requirements: RS_TS_20048

⌈The Time Master and Time Slave shall use the default configuration values as defined by [1, IEEE 802.1 AS] (e.g. MAC destination address or Ethernet frame type), if not otherwise specified within this specification.⌋

[PRS_TS_00005]

Upstream requirements: RS_TS_20048

⌈The Time Master and Time Slave shall start their protocol state machines without Announce message recognition.⌋

[PRS_TS_00206]

Upstream requirements: RS_TS_20048

⌈The Time Master and Time Slave shall ignore the Announce message on the receiver side.⌋

4.2 VLAN Support

[PRS_TS_00163]

Upstream requirements: RS_TS_20048

⌈If `FramePrio` exists, a frame format with priority and VLAN tags shall be used. Otherwise a frame format without priority and VLAN tags shall be used.⌋

4.3 Message format

Some message extensions to the [1, IEEE 802.1 AS] are required. This is accomplished by a new AUTOSAR specific *TLV*, which is using a new IEEE CID (`0x1A75FB`) belonging to AUTOSAR only. An IEEE 802.1AS *TLV* is only available for the `messagetype Announce` (not considered by this specification) and `Follow_Up` (extended by this specification). The `organizationId` of the new *TLV* identifies the AUTOSAR *TLV*, which is succeeding the IEEE 802.1AS *TLV*.

According to [4, IEEE 1588] a Non-AUTOSAR aware switch is supposed to not propagate the AUTOSAR TLV as this TLV is not supported by a Non-AUTOSAR aware switch.

The AUTOSAR *TLV* contains *Sub-TLVs* which always consist of a Type, a Length and a data area.

The usage of the *CRC* is optional. To ensure a great variability between several time observing units, the configuration decides of how to handle the *CRC* of a secured *Sub-TLV*. If the receiver does not support the *CRC* calculation, it might be possible, that a receiver just uses the given values, without evaluating the *CRC* itself.

If the *CRC* option is used, one side effect must be considered. Due to the fact, that `Pdelay` messages do not contain any *TLV*, a *CRC* protection of the related timestamps is not possible. If applications using a *CRC* for `Follow_Up` together with a non-static `Pdelay`, unprotected `Pdelay` time values have to be mixed with protected `Follow_Up` time values, while calculating the value of the corresponding Time Base.

The usage of the *ICV* is optional. To ensure a great variability between several time observing units, the configuration decides on how to handle the *ICV* of a authenticated *Sub-TLV*. If the receiver does not support the *ICV* verification, it might be possible, that a receiver just uses the given values, without verifying the *ICV* itself.

If the *ICV* option is used, then one side effect must be considered. Due to the fact, that `Pdelay` messages do not contain any *TLV*, a *ICV* protection of the related timestamps is not possible. If applications using a *ICV* for `Follow_Up` together with a non-static `Pdelay`, unprotected `Pdelay` time values have to be mixed with protected `Follow_Up` time values, while calculating the value of the corresponding Time Base.

[PRS_TS_00028]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈The message format, etc. shall be derived from [1, IEEE 802.1 AS] chapter 10. Media-independent layer specification and chapter 11. Media-dependent layer specification for full-duplex, point-to-point links, if not otherwise specified.⌋

[PRS_TS_00181]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈The byte order for multibyte values is Big Endian, which is equal to the byte order defined by [1, IEEE 802.1 AS].⌋

4.3.1 Header format

4.3.1.1 Sync and Follow_Up acc. to IEEE 802.1AS

[PRS_TS_00061]

Upstream requirements: RS_TS_20048

⌈If MessageCompliance is set to TRUE, Sync and Follow_Up format shall be supported acc. to [1, IEEE 802.1 AS].⌋

Note: This implies one Time Domain (0).

The table below gives an overview, how an [1, IEEE 802.1 AS] conformant Sync looks like.

Sync Message Header [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
transportSpecific	**messagetype**	1	0	0x10
reserved	**versionPTP**	1	1	0x02
messageLength		2	2	44
domainNumber		1	4	(UInteger8) domainNumber = 0
reserved		1	5	0
flags		2	6	Octet 0: 0x02, Octet 1: 0x00
correctionField		8	8	(Integer64) correctionField
reserved		4	16	0
sourcePortIdentity		10	20	(PortIdentity) portIdentity from origin Time Aware End Station
sequenceId		2	30	(UInteger16) SyncSequenceId =(UInteger16) (prevSyncSequenceId+1)
control		1	32	0
logMessageInterval		1	33	(Integer8) currentLogSyncInterval
Sync Message Fields [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
PTP Message Header		34	0	[refer Sync Message Header]
reserved		10	34	0

Table 4.1: Sync Message Header [IEEE 802.1AS]

The table below gives an overview, how an [1, IEEE 802.1 AS] conformant `Follow_Up` looks like.

Follow_Up Message Header [IEEE 802.1AS]

Follow_Up Message Header [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
transportSpecific	**messagetype**	1	0	`0x18`
reserved	**versionPTP**	1	1	`0x02`
messageLength		2	2	`76`
domainNumber		1	4	`(UInteger8) domainNumber = 0`
reserved		1	5	`0`
flags		2	6	`Octet 0: 0x00, Octet 1: 0x00`
correctionField		8	8	0..281474976710655ns (1ns = 2ˆ16 = 0x0000 0000 0001 0000)
reserved		4	16	`0`
sourcePortIdentity		10	20	`(PortIdentity) portIdentity` from origin Time Aware End Station
sequenceId		2	30	`UInteger16) SyncSequenceId`
control		1	32	`2`
logMessageInterval		1	33	`(Integer8) currentLogSyncInterval`
Follow_Up Message Fields [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
PTP Message Header		34	0	[refer `Follow_Up` Message Header]
preciseOriginTimestamp		10	34	(Timestamp) preciseOriginTimestamp
Follow_Up information TLV		32	44	refer `Follow_Up` information TLV
Follow_Up information TLV [IEEE 802.1AS]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
tlvType		2	0	3
lengthField		2	2	28
organizationId		3	4	0x0080c2
organizationSubType		3	7	1

▽

△

cumulativeScaledRateOffset	4	10	(Integer32)((RateRatio-1) * 2 ˆ41)
gmTimeBaseIndicator	2	14	0
lastGmPhaseChange	12	16	0
scaledLastGmFreqChange	4	28	0

Table 4.2: Follow_Up Message Header [IEEE 802.1AS]

4.3.1.2 Sync and Follow_Up acc. to AUTOSAR

[PRS_TS_00062]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If `MessageCompliance` is set to `FALSE`, the `Sync` and `Follow_Up` format shall be supported acc. to: `Follow_Up Message Header [AUTOSAR]` and `Sync Message Header [AUTOSAR]` depending on configuration.⌋

[PRS_TS_00063]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062, RS_TS_20072

⌈If `MessageCompliance` is set to `FALSE`, the `Follow_Up` shall contain an AUTOSAR *TLV*, depending on configuration.⌋

Message Header [AUTOSAR]

Sync Message Header [AUTOSAR]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
transportSpecific	**messagetype**	1	0	`0x10`
reserved	**versionPTP**	1	1	`0x02`
messageLength		2	2	`44`
domainNumber		1	4	`(UInteger8) domainNumber = 0..127`
reserved		1	5	`0`

flags		2	6	`Octet 0: 0x02,`
				`Octet 1: 0x00`

correctionField		8	8	`(Integer64) correctionField`

▽

△

reserved		4	16	0
sourcePortIdentity		10	20	(PortIdentity) portIdentity from origin Time Aware End Station
sequenceId		2	30	(UInteger16) SyncSequenceId = (UInteger16) (prevSyncSequenceId+1)
control		1	32	0
logMessageInterval		1	33	(Integer8) currentLogSyncInterval
Sync Message Fields [AUTOSAR]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
PTP Message Header		34	0	[refer Sync Message Header]
reserved		10	34	0

Table 4.3: Sync Message Header [AUTOSAR]

Follow_Up Message Header [AUTOSAR]				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
transportSpecific	**messagetype**	1	0	0x18
reserved	**versionPTP**	1	1	0x02
messageLength		2	2	76+10+Sum(Sub-TLVs)
domainNumber		1	4	(UInteger8) domainNumber = 0..127
reserved		1	5	0
flags		2	6	Octet 0: 0x00, Octet 1: 0x00
correctionField		8	8	0..281474976710655ns (1ns = 2ˆ16 = 0x0000 0000 0001 0000)
reserved		4	16	0
sourcePortIdentity		10	20	(PortIdentity) portIdentity from origin Time Aware End Station
sequenceId		2	30	(UInteger16) SyncSequenceId
control		1	32	2
logMessageInterval		1	33	(Integer8) currentLogSyncInterval

▽

△

Follow_Up Message Fields [AUTOSAR]				
High Nibble	Low Nibble	Octets	Offset	Value
PTP Message Header		34	0	[referFollow_Up Message Header]
preciseOrigin-Timestamp		10	34	(Timestamp) preciseOriginTimestamp
Follow_Up information TLV		32 + 10 + sum(Sub - TLVs)	44	[refer Follow_Up information TLV]
Follow_Up information TLV [IEEE 802.1AS]				
High Nibble	Low Nibble	Octets	Offset	Value
tlvType		2	0	3
lengthField		2	2	28
organizationId		3	4	0x0080C2 [IEEE 802.1AS]
organizationSub-Type		3	7	1
cumulativeScale-dRateOffset		4	10	(Integer32)((RateRatio-1) * 2 ˆ41)
gmTimeBaseIndi-cator		2	14	0
lastGm-PhaseChange		12	16	0
scaledLastGm-FreqChange		4	28	0
Follow_Up information TLV [AUTOSAR]				
High Nibble	Low Nibble	Octets	Offset	Value
AUTOSAR TLV Header				
tlvType		2	0	3
lengthField		2	0	6 + Sum(*Sub-TLVs*)
organizationId		3	4	0x1A75FB [AUTOSAR]
organizationSub-Type		3	7	0x605676 [BCD coded GlobalTimeEthTSyn]
AUTOSAR Sub-TLV:Time Secured				
High Nibble	Low Nibble	Octets	Offset	Value
Type		1	0	0x28 [Time secured]
Length		1	1	3

▽

△

CRC_Time_Flags		1	2	BitMask 0x01 [messageLength] BitMask 0x02 [domainNumber] BitMask 0x04 [correctionField] BitMask 0x08 [sourcePortIdentity] BitMask 0x10 [sequenceId] BitMask 0x20 [preciseOriginTimestamp] BitMask 0x40 [reserved] BitMask 0x80 [reserved]
CRC_Time_0		1	3	0..255
CRC_Time_1		1	4	0..255
AUTOSAR Sub-TLV:Status Secured				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
Type		1	0	0x50 [Status secured]
Length		1	1	2
Status		1	2	BitMask 0x01 [SGW with SyncToGTM = 0 SyncToSubDomain = 1] BitMask 0x02 [reserved] BitMask 0x04 [reserved] BitMask 0x08 [reserved] BitMask 0x10 [reserved] BitMask 0x20 [reserved] BitMask 0x40 [reserved] BitMask 0x80 [reserved]
CRC_Status		1	3	0..255
AUTOSAR Sub-TLV:Status Not Secured				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
Type		1	0	0x51 [Status Not Secured]
Length		1	1	2

▽

△

Status		1	2	BitMask 0x01 [SGW with SyncToGTM = 0 SyncToSubDomain = 1] BitMask 0x02 [reserved] BitMask 0x04 [reserved] BitMask 0x08 [reserved] BitMask 0x10 [reserved] BitMask 0x20 [reserved] BitMask 0x40 [reserved] BitMask 0x80 [reserved]
reserved		1	3	0
AUTOSAR Sub-TLV:UserData Secured				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
Type		1	0	0x60 [UserData secured]
Length		1	1	5
UserDataLength		1	2	0..3
UserByte_0		1	3	0..255 (default: 0)
UserByte_1		1	4	0..255 (default: 0)
UserByte_2		1	5	0..255 (default: 0)
CRC_UserData		1	6	0..255
AUTOSAR Sub-TLV:UserData Not Secured				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
Type		1	0	0x61 [UserData not secured]
Length		1	1	5
UserDataLength		1	2	0..3
UserByte_0		1	3	0..255 (default: 0)
UserByte_1		1	4	0..255 (default: 0)
UserByte_2		1	5	0..255 (default: 0)
reserved		1	6	0
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
AUTOSAR Sub-TLV:Time Authenticated				
High Nibble	**Low Nibble**	**Octets**	**Offset**	**Value**
Type		1	0	0x70 [Time Authenticated]
Length		1	1	2..216

▽

△

ICV_Flags	1	2	BitMask 0x01 [ICV with FV] BitMask 0x02 [ICV generation failed] BitMask 0x04 [ICV in multiple Sub-TLV] BitMask 0x08 [reserved] BitMask 0x10 [reserved] BitMask 0x20 [reserved] BitMask 0x40 [reserved] BitMask 0x80 [reserved]
SequenceNumber	1	3	0..4 Sequence number of Sub-TLV:Time Authenticated
FreshnessValueLength	1	4	This field is optional. If not present, then bit [ICV with FV] in ICV_Flags is 0. 0..64 Bits
FV	FVL (in Bytes)	5	This field is optional. If not present, then bit [ICV with FV] in ICV_Flags is 0.
ICV	I	4+1+FVL (in Bytes)	0..205 Bytes (Sequence Number is 0) 1..214 Bytes (Sequence Number is greater than 0) The value of I shall represent the number of octets in the field. If the ICV calculation failed, then it shall have the value of 0 octets.

Table 4.4: Follow_Up Message Header [AUTOSAR]

4.3.1.3 Follow_Up Message Header [AUTOSAR]

[PRS_TS_00066]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈The `messageLength` of the `Follow_Up` Message Header has to be adapted according to the length of all existing *TLVs*.⌋

4.3.1.4 AUTOSAR *TLV* Header

[PRS_TS_00067]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈The AUTOSAR *TLV* Header has a multiplicity of 1.⌋

[PRS_TS_00068]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If an AUTOSAR *TLV* Header exists, at least one AUTOSAR or OEM *Sub-TLV* must exist as well.⌋

[PRS_TS_00069]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If an *AUTOSAR TLV* Header exists, the `lengthField` shall be adapted according to the accumulated size of the subsequent *AUTOSAR and OEM Sub-TLVs*.⌋

4.3.1.5 AUTOSAR and OEM Sub-TLVs

In addition to *Sub-TLVs* defined by AUTOSAR it is allowed to also use OEM specific *Sub-TLVs*.

[PRS_TS_00256]

Status: DRAFT

Upstream requirements: RS_TS_20048, RS_TS_20062

⌈OEM *Sub-TLVs* shall have a Type field in the range of 0xA0 to 0xFF. The AUTOSAR Time Synchronization protocol shall reserve this range for OEM specific *Sub-TLVs*.⌋

[PRS_TS_00070]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If an AUTOSAR or *Sub-TLV* exists, it shall be placed after the AUTOSAR *TLV* Header.⌋

[PRS_TS_00071]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062, RS_TS_20072

⌈If more than one AUTOSAR or OEM *Sub-TLV* exists, each *Sub-TLV* shall be placed after the preceding *Sub-TLV* without gaps.⌋

Note: If more than one *Sub-TLV* exists, the position of each *Sub-TLV* is arbitrary except *Sub-TLV*:Time Authenticated. It is assumed that the order of the *Sub-TLVs* does not change during runtime for a given configuration.

[PRS_TS_00257]

Status: DRAFT
Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062, RS_TS_20072

⌈If a *Sub-TLV*:Time Authenticated exists, a Time Master shall place it after the last AUTOSAR *Sub-TLV*.⌋

Note: OEM *Sub-TLVs* can be placed before or after a *Sub-TLV*:Time Authenticated. If being placed after *Sub-TLV*:Time Authenticated the OEM *Sub-TLVs* are not cryptographically protected (refer to [PRS_TS_00238]).

[PRS_TS_00220]

Status: DRAFT
Upstream requirements: RS_TS_20072

⌈All AUTOSAR and OEM *Sub-TLVs* shall have a Type field of length 1 (byte) and a Length field of length 1 (byte).⌋

Rationale:
Length field has been limited to 1 byte for resource efficiency.

4.3.1.6 AUTOSAR *Sub-TLV*:Time Secured

[PRS_TS_00074]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈The AUTOSAR *Sub-TLV*:Time Secured has a multiplicity of 1 and is only available, if *CRC* protection is required.⌋

[PRS_TS_00075]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If `MessageCompliance` is `FALSE` and `TxSubTLVTime` is set to `TRUE`, the Time Master shall send a `Follow_Up`, which contains an AUTOSAR *Sub-TLV*:Time Secured.⌋

4.3.1.7 AUTOSAR *Sub-TLV*:Status Secured / Not Secured

[PRS_TS_00076]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈The AUTOSAR *Sub-TLV*:Status has a multiplicity of 1 and can either be *CRC* protected (Status Secured) or not (Status Not Secured).⌋

[PRS_TS_00077]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If `MessageCompliance` is set to `FALSE` and `TxSubTLVStatus` is set to `TRUE`, the Time Master shall send a `Follow_Up`, which contains an AUTOSAR `Sub-TLV:Status Secured` or `Sub-TLV:Status Not Secured`.⌋

4.3.1.8 AUTOSAR *Sub-TLV*:UserData Secured / Not Secured

[PRS_TS_00078]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈The AUTOSAR *Sub-TLV*:UserData has a multiplicity of 1 and can either be *CRC* protected (UserData Secured) or not (UserData Not Secured).⌋

[PRS_TS_00079]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈If `MessageCompliance` is set to `FALSE` and `TxSubTLVUserData` is set to `TRUE`, the Time Master shall send a `Follow_Up`, which contains an AUTOSAR `Sub-TLV:UserData Secured` or `Sub-TLV:UserData Not Secured`.⌋

4.3.1.9 AUTOSAR *Sub-TLV*:Time Authenticated

[PRS_TS_00221]

Upstream requirements: RS_TS_20072

⌈The AUTOSAR *Sub-TLV*:Time Authenticated shall have a multiplicity of 5.⌋

[PRS_TS_00222]

Upstream requirements: RS_TS_20072

⌈The AUTOSAR *Sub-TLV*:Time Authenticated shall not be *CRC* protected.⌋

[PRS_TS_00223]

Upstream requirements: RS_TS_20072

⌈If MessageCompliance is set to FALSE and TLVFollowUpICVSubTLV is set to TRUE, the Time Master shall send a Follow_Up, which contains the AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00224]

Upstream requirements: RS_TS_20072

⌈The length of the FV field of AUTOSAR *Sub-TLV*:Time Authenticated shall be configurable (GlobalTimeIcvFvLength).⌋

[PRS_TS_00225]

Upstream requirements: RS_TS_20072

⌈The length of the ICV field of AUTOSAR *Sub-TLV*:Time Authenticated shall be configurable (GlobalTimeIcvLength).⌋

[PRS_TS_00226]

Upstream requirements: RS_TS_20072

⌈When ICV value does not fit within one AUTOSAR *Sub-TLV*:Time Authenticated, the Follow_Up message shall contain multiple AUTOSAR *Sub-TLV*:Time Authenticated with fragmented ICV value in each AUTOSAR *Sub-TLV*:Time Authenticated.⌋

Rationale:
Fragmentation of the ICV allows for bigger ICV value, because the length of the value field of a single AUTOSAR *Sub-TLV* is limited to 255 bytes (refer to [PRS_TS_00220]).

[PRS_TS_00227]

Upstream requirements: RS_TS_20072

⌈When Follow_Up message contains multiple AUTOSAR *Sub-TLV*:Time Authenticated, the Time Master shall fragment the ICV value into n (n is less than or equal to 5) fragments.

- The length of first fragment shall not exceed (MAXLEN_SUBTLV_TIMEAUTH - LEN_SUBTLV_TIMEAUTH_PCI - LEN_FVL - FVL) bytes.
- The length of the following fragments shall not exceed (MAXLEN_SUBTLV_TIMEAUTH - LEN_SUBTLV_TIMEAUTH_PCI) bytes.

With
MAXLEN_SUBTLV_TIMEAUTH = 216 (refer to the 'length' field of AUTOSAR *Sub-TLV*:Time Authenticated in [PRS_TS_00063])
LEN_SUBTLV_TIMEAUTH_PCI = 2 (length of 'ICV_Flags' field + length of 'SequenceNumber' field)

LEN_FVL (length of the optional 'FVL' field) = 1, if bit [ICV with FV] of ICV_Flags is set. Otherwise set to 0.
⌋

Note: Follow_Up message (with 1500 bytes of payload) would allow for an ICV length of up to 1061 bytes.

[PRS_TS_00228]

Upstream requirements: RS_TS_20072

⌈When Follow_Up message contains multiple AUTOSAR *Sub-TLV*:Time Authenticated, the Time Master shall put the ICV fragments according to their significance in ascending order into the AUTOSAR *Sub-TLV*:Time Authenticated, i.e., the most significant fragment is contained in AUTOSAR *Sub-TLV*:Time Authenticated with sequence number 0.⌋

[PRS_TS_00229]

Upstream requirements: RS_TS_20072

⌈The Time Master shall set the sequence number of the first AUTOSAR *Sub-TLV*:Time Authenticated in Follow_Up message to 0. When Follow_Up message contains multiple AUTOSAR *Sub-TLV*:Time Authenticated, the Time Master shall increment the sequence number by 1 in the consecutive AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00230]

Upstream requirements: RS_TS_20072

⌈When Follow_Up message contains multiple AUTOSAR *Sub-TLV*:Time Authenticated, the Time Master shall reset the bit 'ICV in multiple Sub-TLV' in ICV_Flags in AUTOSAR *Sub-TLV*:Time Authenticated with the last fragmented ICV value. All other AUTOSAR *Sub-TLV*:Time Authenticated in thatFollow_Up message shall have the bit 'ICV in multiple Sub-TLV' in ICV_Flags set.⌋

[PRS_TS_00231]

Upstream requirements: RS_TS_20072

⌈WhenFollow_Up message contains multiple AUTOSAR *Sub-TLV*:Time Authenticated,

- AUTOSAR *Sub-TLV*:Time Authenticated with the sequence number equal to 0 shall have the FV field included and the FVL field accordingly filled
- AUTOSAR *Sub-TLV*:Time Authenticated with the sequence number not equal to 0 shall not include the FV and FVL field

⌋

[PRS_TS_00232]

Upstream requirements: RS_TS_20072

⌈In the below cases,

- Time Aware Bridge with GTM not as Management CPU
- Time Aware Bridge with switch device running a firmware which provides the Switch Management and Global Time support

the Time Master shall add the AUTOSAR *Sub-TLV*:Time Authenticated with correctionField having value ’0’. And the Time Aware Bridge shall update the AUTOSAR *Sub-TLV*:Time Authenticated with the updated value of correctionField.⌋

[PRS_TS_00233]

Upstream requirements: RS_TS_20072

⌈In the case of cascaded Time Aware Bridges, each bridge shall verify the ICV in the received AUTOSAR *Sub-TLV*:Time Authenticated. If ICV verification is successful, the bridge shall update the AUTOSAR *Sub-TLV*:Time Authenticated after updating the correctionField and CrcCorrectionField in received`Follow_Up` message. If ICV verification fails, the bridge shall discard the received `Follow_Up` message.

⌋

4.3.2 Body/Payload format

Placeholder for upcoming AUTOSAR releases.

4.3.3 Data Types

Refer to [1, IEEE 802.1 AS].

4.4 Message types

Refer to [1, IEEE 802.1 AS].

4.4.1 Data Messages

Refer to [1, IEEE 802.1 AS].

4.4.2 Control Messages

Refer to [1, IEEE 802.1 AS].

4.5 Services / Commands

Placeholder for upcoming AUTOSAR releases.

4.6 Sequences (lower layer)

4.6.1 Pdelay Protocol for Latency Calculation

Figure 4.1 illustrates the Propagation Delay Measurement (Pdelay) sequence using Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up messages as defined in [1, IEEE802.1 AS] chapter 11.1.2 "Propagation delay measurement". Due to the limitation given in chapter 1.2.2 “Limitations”, it is sufficient that only the Time Slave initiates the Pdelay measurement.

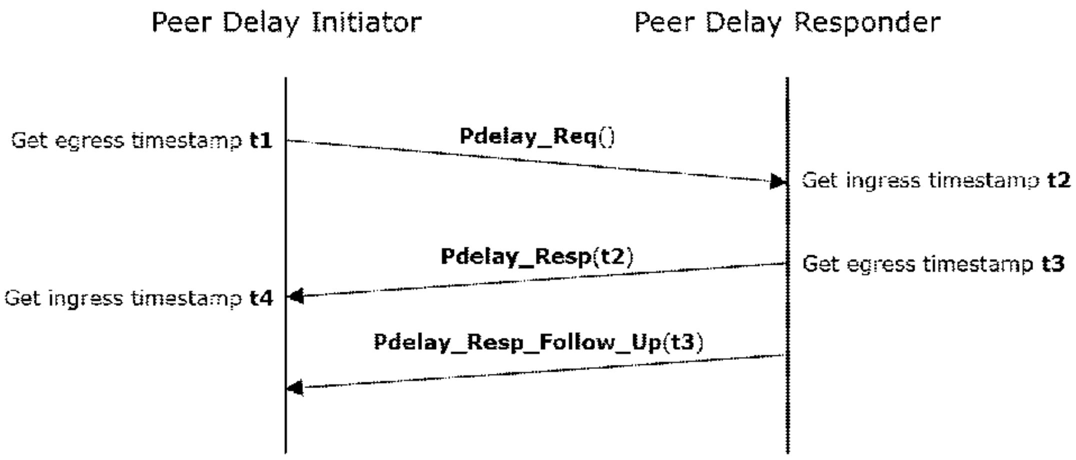

Figure 4.1: Propagation Delay Measurement (Pdelay)

[PRS_TS_00154]

Upstream requirements: RS_TS_20048

⌈If GlobalTimeTxPdelayReqPeriod is not equal to 0 and if the Pdelay latency calculation result exceeds PdelayLatencyThreshold, the measured value shall be discarded and the previous value shall be kept.⌋

[PRS_TS_00219]

Upstream requirements: RS_TS_20048, RS_TS_20051, RS_TS_20071

⌈If

- a Pdelay_Resp message or a Pdelay_Resp_Follow_Up message is received by a Peer Delay Initiator
- and the requestingPortIdentity of the message does not match the sourcePortIdentity of the Peer Delay Requester,

the Peer Delay Initiator shall ignore the received messages.⌋

Rationale: In multidrop topologies (like 10BASE-T1S) a node may receive more than one Pdelay_Resp message and thus even Pdelay_Resp messages for "foreign" Pdelay_Req messages responding to requests from other nodes. To prevent system degradation foreign Pdelay_Resp messages shall be ignored.

[PRS_TS_00004]

Upstream requirements: RS_TS_20048, RS_TS_20051

⌈A Pdelay_Resp timeout or incomplete Pdelay protocol with the exception of [PRS_TS_00219] shall stop the latency calculation algorithm. In such cases, the device shall use the latest successful calculated latency value.⌋

Note: A timeout is detected, when sending the next subsequent Pdelay_Req before receiving the Pdelay_Resp resp. Pdelay_Resp_Follow_Up belonging to the Pdelay_Req before.

[PRS_TS_00164]

Upstream requirements: RS_TS_20048, RS_TS_20051

⌈

If a Pdelay_Req has been transmitted (waiting for Pdelay_Resp) or if a Pdelay_Resp has been received (waiting for Pdelay_Resp_Follow_Up), the Peer Delay Initiator shall observe the Pdelay timeout as given by PdelayRespAndRespFollowUpTimeout. A value of 0 deactivates this timeout observation.

⌋

[PRS_TS_00210]

Upstream requirements: RS_TS_20048, RS_TS_20051

⌈If a reception timeout occurs (refer to [PRS_TS_00164]), any received Pdelay_Resp resp. Pdelay_Resp_Follow_Up shall be ignored, until a new Pdelay_Req has been sent.⌋

[PRS_TS_00265] Initialization of linkDelay with static value

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈When Pdelay Initiator is initialized, it shall set the `linkDelay` value to the static value `GlobalTimePropagationDelay`.⌋

[PRS_TS_00140]

Upstream requirements: RS_TS_20066

⌈If `GlobalTimeTxPdelayReqPeriod` equals 0, the `pDelay` Initiator shall not measure the propagation delay. The Time Slave shall use a static value `GlobalTimePropagationDelay` as propagation delay instead.⌋

Note: Since `GlobalTimeTxPdelayReqPeriod` is ECU specific, neither a Time Master nor all Time Slaves have to measure the propagation delay. Global Time Synchronization in AUTOSAR does yet not define dynamic reconfiguration or backup strategies that will reassign the role as Time Master, therefore propagation delay measurements make currently no sense for a Time Master (although a Time Master shall be able to handle `Pdelay_Req` initiated by a Time Slave).

[PRS_TS_00003]

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If `GlobalTimeTxPdelayReqPeriod` is set to 0, the Peer Delay Initiator shall calculate the value of the value `linkDelay` according to [PRS_TS_00264],⌋

Note: If `GlobalTimeTxPdelayReqPeriod` is not 0, the Time Sync module does a Propagation Delay (Pdelay) Measurement according to [1, IEEE802.1 AS] chapter 11.1.2 "Propagation delay measurement" (refer also to [PRS_TS_00141]).

[PRS_TS_00264]

Status: DRAFT
Upstream requirements: RS_TS_20048, RS_TS_20066

⌈When a valid `Pdelay_Resp_Follow_Up` message is received and a new `neighborRateRatio` has been calculated,

then a Peer Delay Initiator shall calculate the link delay for the link according to the following formula:

$$linkDelay = rateRatio_{PdelayResponder} * \frac{neighborRateRatio * (t4 - t1) - (t3 - t2)}{2} \quad (4.1)$$

With

- rateRatio$_{\text{PdelayResponder}}$ as calculated according to [PRS_TS_00262]
- and `neighborRateRatio` as calculated according to [PRS_TS_00259]

⌋

Note: The linkDelay is calculated relative to the time base of the Global Time Master. The mean propagation delay, i.e.,

$$\frac{neighborRateRatio * (t4 - t1) - (t3 - t2)}{2} \tag{4.2}$$

which is defined by [1, IEEE 802.1 AS], chapter 10.2.4.7 "neighborPropDelay" and 11.2.15.2.4 "computePropTime" is the link delay measured based on local clock of the Peer Delay Responder. Multiplication by $\mathtt{rateRatio}_{\mathtt{PdelayResponder}}$ as in Equation 4.1 above converts it to the time base of the Global Time Master.

[PRS_TS_00149]

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If GlobalTimeTxPdelayReqPeriod is greater than 0, the Peer Delay Initiator shall cyclically measure the propagation delay only on that Time Domain with the lowest Time Domain ID and shall use this value to adjust all corresponding Time Bases.⌋

Note: There is no need to measure the propagation delay for all Time Domains, because the same value is expected. This requirement ensures also the usage of Time Domain 0 for Pdelay, to be compatible to [1, IEEE 802.1 AS].

[PRS_TS_00142]

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If GlobalTimeTxPdelayReqPeriod is greater than 0,GlobalTimePropagationDelay shall be used as default value for the propagation delay, until first valid propagation delay has been measured.⌋

[PRS_TS_00011]

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If GlobalTimeTxPdelayReqPeriod is greater than 0, the Peer Delay Initiator shall periodically transmit Pdelay_Req for latency calculation with the cycle GlobalTimeTxPdelayReqPeriod as defined in [1, IEEE 802.1 AS] chapter 11.1.2 "Propagation delay measurement".⌋

Note: GlobalTimePdelayRespEnable allows disabling of Pdelay_Resp and Pdelay_Resp_Follow_Up, if no Pdelay_Req is expected to be received, i.e. for the Time Master, if all Time Slaves have set GlobalTimeTxPdelayReqPeriod to 0 or for any Time Slave if the Time Master has set GlobalTimeTxPdelayReqPeriod to 0.

[PRS_TS_00012]

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If GlobalTimePdelayRespEnable is set to TRUE, the Peer Delay Responder shall react to Pdelay_Req by transmitting Pdelay_Resp for latency calculation as defined in [1, IEEE 802.1 AS] chapter 11.1.2 "Propagation delay measurement".⌋

[PRS_TS_00143]

Upstream requirements: RS_TS_20066

⌈If GlobalTimePdelayRespEnable is set to FALSE, Pdelay_Resp and Pdelay_Resp_Follow_Up shall be omitted.⌋

[PRS_TS_00141]

Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If GlobalTimeTxPdelayReqPeriod is greater than 0, the Peer Delay Initiator shall cyclically measure the propagation delay using Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up as defined in [1, IEEE802.1 AS] chapter 11.1.2 "Propagation delay measurement".⌋

4.6.2 Rate Ratio Calculation

Based on the ingress and egress timestamps t3 and t4 as given in Figure 4.1 a Peer Delay Initiator is able to calculate the neighborRateRatio. neighborRateRatio is the ratio of the frequency of the local clock of the Peer Delay Responder to the frequency of the local clock of the Peer Delay Initiator.

[PRS_TS_00259]

Status: DRAFT
Upstream requirements: RS_TS_20075

⌈If RateRatioEnable is set to TRUE

when a Pdelay_Resp_Follow_Up message is received,

a Peer Delay Initiator shall calculate the current value of the neighborRateRatio across previous N successive, successful Pdelay measurements according to [1, IEEE 802.1 AS], chapter 11.2.15.2.3 computePdelayRateRatio(), using the following formula:

$$neighborRateRatio = \frac{t3_i - t3_{(i-N)}}{t4_i - t4_{(i-N)}} \tag{4.3}$$

With

- N: number of Pdelay measurements used for calculation as given by the configuration parameter RateRatioMeasurementCount
- t3$_i$, t3$_{i-N}$: egress timestamps of the Pdelay_Resp messages on Peer Delay Responder side as received in the Pdelay_Resp_Follow_Up messages by the Peer Delay Initiator belonging to the current, i.e., ith and the (i-N)th Pdelay measurement, respectively (see figure referenced in Note below)
- t4$_i$, t4$_{i-N}$: ingress timestamps of the Pdelay_Resp messages on Peer Delay Initiator side belonging to the current, i.e., ith and the (i-N)th Pdelay measurement, respectively (see figure referenced in Note below)

If RateRatioEnable is set to FALSE

a Peer Delay Initiator shall set the neighborRateRatio to 1

⌋

Note: Figure 4.1 “Propagation Delay Measurement (Pdelay)”

[PRS_TS_00260]

Status: DRAFT
Upstream requirements: RS_TS_20075

⌈If

- RateRatioEnable is set to TRUE
- and no neighborRateRatio has yet been calculated

then a Peer Delay Initiator shall set the neighborRateRatio value to 1.⌋

Based on the calculated neighborRateRatio and the cumulativeScaledRateOffset value as received in the Follow-Up message a Time Slave/Time Gateway can derive the rateRatio, which is the ratio of the frequency of Global Time Master to the frequency of the local clock of the Time Slave/Time Gateway

[PRS_TS_00261]

Status: DRAFT
Upstream requirements: RS_TS_20075

⌈If RateRatioEnable is set to TRUE,

when a valid Follow-Up message is received and a new neighborRateRatio has been calculated,

a Time Slave and a Time Gateway shall calculate the rateRatio as

$$rateRatio = rateRatio_{PdelayResponder} + (neighborRateRatio - 1.0) \tag{4.4}$$

With

- $rateRatio_{PdelayResponder}$ as calculated according to [PRS_TS_00262]
- and neighborRateRatio as calculated according to [PRS_TS_00259]

⌋

[PRS_TS_00262]

Status: DRAFT
Upstream requirements: RS_TS_20048, RS_TS_20066

⌈If RateRatioEnable is set to TRUE,

when a valid Pdelay_Resp_Follow_Up message is received,

then a Peer Delay Initiator shall calculate the value that represents the rateRatio of the Peer Delay Responder to the Global Time Master $rateRatio_{PdelayResponder}$ as

$$rateRatio_{PdelayResponder} = \left(cumulativeScaledRateOffset/2^{41} + 1.0\right) \tag{4.5}$$

If RateRatioEnable is set to FALSE,

then a Peer Delay Initiator shall set $rateRatio_{PdelayResponder}$ to 1.⌋

[PRS_TS_00263]

Status: DRAFT
Upstream requirements: RS_TS_20075

⌈If RateRatioEnable is set to TRUE, a Time Gateway and a Time-aware Bridge shall calculate the value cumulativeScaledRateOffset according to [1, IEEE 802.1 AS], chapter 11.4.4.3.6 "cumulativeScaledRateOffset (Integer32)" as

$$cumulativeScaledRateOffset = (rateRatio - 1.0) * 2^{41} \tag{4.6}$$

and shall truncate the calculated value to the next smaller integer.

With

- rateRatio as calculated according to [PRS_TS_00261]

A Time Gateway and a Time-aware Bridge shall forward the truncated cumulativeScaledRateOffset value in the Follow-Up message.⌋

4.6.3 Acting as Time Master

A Time Master is an entity which is the master for a certain Time Base and which propagates this Time Base to a set of Time Slaves within a certain segment of a communication network, being a source for this Time Base.

If a Time Master is also the owner of the Time Base then he is the Global Time master. A Time Gateway typically consists of one Time Slave and one or more Time Masters.

When mapping time entities to real ECUs, an ECU could be Time Master (or even Global Time Master) for one Time Base and Time Slave for another Time Base.

4.6.3.1 Message Processing

[PRS_TS_00050]

Upstream requirements: RS_TS_20047, RS_TS_20048

⌈The Time Master shall support the transmission of `Sync` and `Follow_Up` according to [1, IEEE 802.1 AS] as well as the transmission and reception of `Pdelay_Req`, `Pdelay_Resp` and `Pdelay_Resp_Follow_Up`.⌋

[PRS_TS_00016]

Upstream requirements: RS_TS_20047, RS_TS_20048

⌈

- `GLOBAL_TIME_BASE` bit within the `timeBaseStatus`, which is read from the corresponding Time Base, is set

and

- `GlobalTimeTxPeriod` is not 0.

⌋

[PRS_TS_00018]

Upstream requirements: RS_TS_20048

⌈The `preciseOriginTimestamp` as calculated above, shall be used in the transmission of the `Follow_Up` as defined in [1, IEEE 802.1 AS] chapter 11.1.3 "Transport of time-synchronization information".⌋

4.6.3.1.1 Frame Debouncing

[PRS_TS_00186]

Upstream requirements: RS_TS_20047

⌈If multiple frames are triggered at the same time, the frames shall be sent in the following order:

1. `Sync`
2. `Follow_Up`
3. `Pdelay_Req`

4. Pdelay_Resp, Pdelay_Resp_Follow_Up

⌋

4.6.3.2 Message Field Calculation and Assembling

[PRS_TS_00092]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈If MessageCompliance is set to FALSE, a Time Master shall add an *AUTOSAR TLV* to the Follow_Up frame.⌋

[PRS_TS_00091]

Upstream requirements: RS_TS_20061

⌈If MessageCompliance is set to FALSE, CRC_SUPPORT shall be considered.⌋

[PRS_TS_00093]

Upstream requirements: RS_TS_20061, RS_TS_20072

⌈Depending on CRC_SUPPORT the Follow_Up.TLV[AUTOSAR].Sub-TLV.Type shall be:⌋

Follow_Up Message Header [IEEE 802.1AS]

	Sub-TLV.Type	
GlobalTimeTxCrcSecured	CRC_SUPPORTED	CR_NOT_SUPPORTED
	0x28 *Sub-TLV*:Time Secured is *CRC* secured	n.a.
	0x50 *Sub-TLV*:Status is *CRC* secured	0x51 *Sub-TLV*:Status is not *CRC* secured
	0x60 *Sub-TLV*:UserData is *CRC* secured	0x61 *Sub-TLV*:UserData is not *CRC* secured
	0x70 *Sub-TLV*:Time Authenticated is not *CRC* secured	0x70 *Sub-TLV*:Time Authenticated is not *CRC* secured

4.6.3.2.1 SGW Calculation

[PRS_TS_00094]

Upstream requirements: RS_TS_20052, RS_TS_20054

⌈The *SGW* value (Time Gateway synchronization status) shall be mapped to the Status element of the *AUTOSAR Sub-TLV*:Status.

If the `SYNC_TO_GATEWAY` is set, the `SGW` value shall be `SyncToSubDomain`. Otherwise, it shall be `SyncToGTM`.⌋

4.6.3.2.2 CRC Calculation

[PRS_TS_00266] CRC Calculation of Time Master

Upstream requirements: RS_TS_20061

⌈The *CRC* calculation of the Time Master shall use the generator polynomial *0x2F*, the initial value *0xFF* and the *XOR* value *0xFF*. Neither the input data nor the result data shall be reflected.⌋

Note: The *CRC* calculation is based on the AUTOSAR E2E Profile 2. For details refer to [5, FO-PRS-E2EProtocol].

[PRS_TS_00097]

Upstream requirements: RS_TS_20061

⌈The `DataID` shall be calculated as: `DataID = DataIDList[Follow_Up.sequenceId mod 16]`, where `DataIDList` is given by configuration for the `Follow_Up`.⌋

Note: A specific `DataID` out of a predefined `DataIDList` ensures the identification of data elements of Time Synchronization messages.

[PRS_TS_00182]

Upstream requirements: RS_TS_20061

⌈If applying the *CRC* calculation on multibyte values, the byte order shall be such, that the byte containing the most significant bit of the value shall be used first.⌋

[PRS_TS_00184]

Upstream requirements: RS_TS_20061

⌈If applying the *CRC* calculation on multibyte message data, the byte order shall be in ascending order of the octets, i.e., the octet with the lowest offset shall be used first.⌋

4.6.3.2.2.1 AUTOSAR *Sub-TLV*:Time Secured

[PRS_TS_00098]

Upstream requirements: RS_TS_20061

⌈If GlobalTimeTxCrcSecured is CRC_SUPPORTED, the Time Master shall write the contents of CrcTimeFlagsTxSecured to CRC_Time_Flags acc. to the following rule.⌋

	CrcTimeFlagsTxSecured contents:	
CRC_Time_Flags	Follow_Up Message Header	Follow_Up Message Field
BitMask 0x01	CRCMessageLength	n.a.
BitMask 0x02	CRCDomainNumber	n.a.
BitMask 0x04	CrcCorrectionField	n.a.
BitMask 0x08	CRCSourcePortIdentity	n.a.
BitMask 0x10	CRCSequenceIdentity	n.a.
BitMask 0x20	n.a.	CRCPrecise - OriginTimestamp
BitMask 0x40	n.a.	n.a.
BitMask 0x80	n.a.	n.a.

[PRS_TS_00099]

Upstream requirements: RS_TS_20061

⌈If GlobalTimeTxCrcSecured is CRC_SUPPORTED, the Time Master shall calculate the *CRC* for CRC_Time_0 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in CrcTimeFlagsTxSecured acc. to the rule in the table below and the DataID. The data elements used for the calculation of the CRC shall apply the following order:

1. the value of CRC_Time_Flags
2. the domainNumber inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask *0x02*
3. the sourcePortIdentity inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask *0x08*

4. the preciseOriginTimestamp inside the Follow_Up Message Field, if CRC_Time_Flags contains BitMask *0x20*

5. the DataID

⌋

	For CRC_Time_0 calculation considered contents:	
If CRC_Time_Flags is set to 1	Follow_Up Message Header	Follow_Up Message Field
BitMask 0x01	n.a.	n.a.
BitMask 0x02	domainNumber	n.a.
BitMask 0x04	n.a.	n.a.
BitMask 0x08	sourcePortIdentity	n.a.
BitMask 0x10	n.a.	n.a.
BitMask 0x20	n.a.	preciseOriginTimestamp
BitMask 0x40	n.a.	n.a.
BitMask 0x80	n.a.	n.a.

Note: CRC_Time_Flags is having the same value like the configuration item CrcTimeFlagsTxSecured, whereas the resulting *CRC* of the dependent items remains network wide unchanged.

[PRS_TS_00100]

Upstream requirements: RS_TS_20061

⌈If GlobalTimeTxCrcSecured is set to CRC_SUPPORTED, the Time Master shall calculate the CRC for CRC_Time_1 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in CrcTimeFlagsTxSecured acc. to the rule in the table below and the DataID. The data elements used for the calculation of the *CRC* shall apply the following order:

1. the value of CRC_Time_Flags

2. the messageLength inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x01

3. the correctionField inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x04

4. the sequenceId inside the Follow_Up Message Header, if CRC_Time_Flags contains BitMask 0x10

5. the DataID

⌋

	For CRC_Time_1 calculation considered contents:	
If CRC_Time_Flags is set to 1	Follow_Up Message Header	Follow_Up Message Field
BitMask 0x01	messageLength	n.a.
BitMask 0x02	n.a.	n.a.
BitMask 0x04	correctionField	n.a.
BitMask 0x08	n.a.	n.a.
BitMask 0x10	sequenceId	n.a.
BitMask 0x20	n.a.	n.a.
BitMask 0x40	n.a.	n.a.
BitMask 0x80	n.a.	n.a.

Note: CRC_Time_Flags has the same value as the configuration item CrcTimeFlagsTxSecured.

4.6.3.2.2.2 AUTOSAR Sub-TLV:Status secured

[PRS_TS_00101]

Upstream requirements: RS_TS_20061

⌈If GlobalTimeTxCrcSecured is set to CRC_SUPPORTED, the Time Master shall calculate the CRC for CRC_Status by considering the contents of Status and DataID (in this order).⌋

4.6.3.2.2.3 AUTOSAR Sub-TLV:UserData secured

[PRS_TS_00102]

Upstream requirements: RS_TS_20061

⌈If GlobalTimeTxCrcSecured is set to CRC_SUPPORTED, the Time Master shall calculate the CRC for CRC_UserData by considering the contents of UserDataLength, UserByte_0, UserByte_1, UserByte_2 and DataID (in this order).⌋

4.6.3.2.3 Sequence Counter (sequenceId) Calculation

[PRS_TS_00187]

Upstream requirements: RS_TS_20061

⌈The Sequence Counter (sequenceId) of a Sync and Pdelay_Req message shall be initialized with 0.⌋

[PRS_TS_00188]

Upstream requirements: RS_TS_20061

⌈The Peer Delay Initiator shall increment the Sequence Counter of a Pdelay_Req message by 1 on each transmission request for a Pdelay_Req message. The Sequence Counter shall wrap around at 65535 to 0 again.⌋

[PRS_TS_00189]

Upstream requirements: RS_TS_20061

⌈The Time Master shall increment the Sequence Counter of aSync message by 1 on each transmission request for a Sync message of a given Time Domain. The Sequence Counter shall wrap around at 65535 to 0 again.⌋

[PRS_TS_00190]

Upstream requirements: RS_TS_20061

⌈The Time Master shall set the Sequence Counter (sequenceId) value for a Follow_Up message to the Sequence Counter (sequenceId) value of the corresponding Sync message.⌋

[PRS_TS_00191]

Upstream requirements: RS_TS_20061

⌈The Peer Delay Responder shall set the Sequence Counter (sequenceId) value for a Pdelay_Resp and Pdelay_Resp_Follow_Up message to the Sequence Counter (sequenceId) value of the corresponding Pdelay_Req message.⌋

4.6.3.2.4 ICV Generation

Each timebase is configured with at least one Freshness Value (FV). The FV refers to a monotonic counter that is used to ensure freshness of the timebase. Such a monotonic counter could be realized by means of individual message counters, called Freshness Counter, or by a time stamp value called Freshness Timestamp.

The ICV refers to the result of a cryptographic function, that are used to ensure that unauthorized modifications of a message are detected. A cryptographic function can be of any primitive with the associated cryptographic key.

[PRS_TS_00234]

Upstream requirements: RS_TS_20072

⌈When (GlobalTimeIcvFvLength) is configured greater than 0, then the Time Master shall derive the FV and include the FV in the ICV generation.⌋

[PRS_TS_00235]

Upstream requirements: RS_TS_20072

⌈When (GlobalTimeIcvFvLength) is configured greater than 0, then the Time Master shall add the FV, the length of FV (FVL) and set the 'FV in ICV' flag of ICV_Flags in AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00236]

Upstream requirements: RS_TS_20072

⌈When (GlobalTimeIcvFvLength) is configured to 0, then the Time Master shall not add the FV, set the length of FV (FVL) to 0 and reset the 'FV in ICV' flag of ICV_Flags in AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00237]

Upstream requirements: RS_TS_20072

⌈When (GlobalTimeIcvFvLength) is configured greater than 0 and the Time Master fails to derive the FV, then the ICV generation shall be considered as failed. In this case, the Time Master shall reset the 'FV in ICV' and set the 'ICV generation failed' flags of ICV_Flags in AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00238]

Upstream requirements: RS_TS_20072

⌈If TLVFollowUpICVSubTLV is set to TRUE, the Time Master shall generate the ICV value by applying the cryptographic primitive (GlobalTimeIcvCryptoPrimitive) to the content of the Follow_Up message (i.e., the header, the message fields and all TLVs - except for the ICV value itself in the AUTOSAR *Sub-TLV*:Time Authenticated and any OEM *Sub-TLVs* following the AUTOSAR *Sub-TLV*:Time Authenticated).⌋

[PRS_TS_00239]

Upstream requirements: RS_TS_20072

⌈If the ICV generation (including deriving the FV) fails or takes longer than the timeout IcvGenerationTimeout, the Time Master shall set flag 'ICV Generation Failed' in the ICV_Flags field of AUTOSAR *Sub-TLV*:Time Authenticated⌋

[PRS_TS_00240]

Upstream requirements: RS_TS_20072

⌈When ICV value does not fit within one AUTOSAR *Sub-TLV*:Time Authenticated, the Time Master shall fragment the ICV value correctly into multiple AUTOSAR *Sub-TLV*:Time Authenticated (refer to [PRS_TS_00227], [PRS_TS_00228], [PRS_TS_00229], [PRS_TS_00230], [PRS_TS_00231]).⌋

4.6.3.2.5 Message Assembling

[PRS_TS_00104]

Upstream requirements: RS_TS_20048, RS_TS_20061, RS_TS_20062

⌈For each transmission of a Time Synchronization message, the Time Synchronization module shall set-up the message as follows:

1. Assemble Message Header
2. If `Follow_Up`: Calculate `Follow_Up.preciseOriginTimestamp`
3. If `Follow_Up`: Assemble IEEE `TLV`
4. If `Follow_Up`: Assemble AUTOSAR `TLV` (configuration dependent) except the AUTOSAR *Sub-TLV*:Time Authenticated.
5. If `Follow_Up`: Assemble AUTOSAR *Sub-TLV*:Time Authenticated (configuration dependent).

⌋

Note: Section 4.6.3.2.4 provides more details how the `Follow_Up` message shall assemble the AUTOSAR *Sub-TLV*:Time Authenticated.

4.6.4 Acting as Time Slave

A Time Slave is an entity, which is the recipient for a certain Time Base within a certain segment of a communication network, being a consumer for this Time Base .

4.6.4.1 Message processing

[PRS_TS_00023]

Upstream requirements: RS_TS_20048

⌈The Time Slave shall support the reception of `Sync` and `Follow_Up` according [1, IEEE 802.1 AS] as well as the transmission and reception of `Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up,` [PRS_TS_00140], [PRS_TS_00141],[PRS_TS_00004].⌋

[PRS_TS_00025]

Upstream requirements: RS_TS_20048, RS_TS_20051

⌈For each configured Time Slave the Ethernet module shall observe the reception timeout `GlobalTimeFollowUpTimeout` between the `Sync` and its `Follow_Up`.
If no `Follow_Up` received before the reception timeout expires, the Time Slave shall reset the sequence (i.e. waiting for a new Sync).
A value of 0 deactivates this timeout observation.⌋

[PRS_TS_00241]

Upstream requirements: RS_TS_20072

⌈While `GlobalTimeFollowUpTimeout` is running, if the `Sync` message is received, the Time Slave shall discard the `Sync` and shall reset the sequence (i.e. waiting for a new `Sync`).⌋

Note: The general timeout monitoring for the Time Base update is located in the Implementation of Time Synchronization and not in the provider modules.

4.6.4.1.1 Frame Debouncing

[PRS_TS_00242]

Upstream requirements: RS_TS_20047, RS_TS_20072

⌈During `rx_debounce_time` any`Sync` or `Follow_Up` message received shall be discarded and the sequence shall be reset (i.e., waiting for a new `Sync`).⌋

Rationale: Intention of [PRS_TS_00241] and [PRS_TS_00242] is to improve robustness of the Time Synchronization protocol against message sequence errors, specifically injection of fake `Sync` messages by an attacker. Note that this will not allow to filter out all possible fake `Sync` scenarios.

4.6.4.2 Message Field Validation and Disassembling

[PRS_TS_00105]

Upstream requirements: RS_TS_20061, RS_TS_20062, RS_TS_20072

⌈If MessageCompliance is set to FALSE, RxCrcValidated, RxIcvVerification shall be considered.⌋

[PRS_TS_00106]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈If MessageCompliance is set to FALSE, a Time Slave shall check if an AUTOSAR *TLV* in the Follow_Up message exists.⌋

[PRS_TS_00107]

Upstream requirements: RS_TS_20061, RS_TS_20072

⌈The *CRCs* inside the *AUTOSAR TLV* shall be validated, depending on RxCrcValidated and the Follow_Up.TLV[AUTOSAR].Sub-TLV.Type acc. to:⌋

	Sub-TLV.Type	
RxCrcValidated	CRC_VALIDATED	CRC_NOT_VALIDATED
	0x28 Sub-TLV:Time Secured is CRC secured	n.a.
	0x50 Sub-TLV:Status is CRC secured	0x51 Sub-TLV:Status is not CRC secured
	0x60 Sub-TLV:UserData is CRC secured	0x61 Sub-TLV:UserData is not CRC secured
	0x70 Sub-TLV:Time Authenticated is not CRC secured	0x70 Sub-TLV:Time Authenticated is not CRC secured

[PRS_TS_00108]

Upstream requirements: RS_TS_20061, RS_TS_20072

⌈The *CRCs* inside the *AUTOSAR TLV* shall be ignored, if RxCrcValidated is set to CRC_IGNORED and the Follow_Up.TLV[AUTOSAR].Sub-TLV.Type contains any of the following defined values:⌋

	Sub-TLV.Type	
RxCrcValidated	CRC_IGNORED	
	0x28 Sub-TLV:Time Secured is CRC secured	n.a.
	0x50 Sub-TLV:Status is CRC secured	0x51 Sub-TLV:Status is not CRC secured
	0x60 Sub-TLV:UserData is CRC secured	0x61 Sub-TLV:UserData is not CRC secured
	0x70 Sub-TLV:Time Authenticated is not CRC secured	0x70 Sub-TLV:Time Authenticated is not CRC secured

[PRS_TS_00109]

Upstream requirements: RS_TS_20061, RS_TS_20072

⌈The *CRCs* inside the *AUTOSAR TLV* shall be either validated or not validated, if RxCrcValidated is set to CRC_OPTIONAL and the Follow_Up.TLV[AUTOSAR].Sub-TLV.Type contains any of the following defined values:⌋

	Sub-TLV.Type	
RxCrcValidated	CRC_OPTIONAL	
	CRC shall be validated	CRC shall not be validated
	0x28 Sub-TLV:Time Secured is CRC secured	n.a.
	0x50 Sub-TLV:Status is CRC secured	0x51 Sub-TLV:Status is not CRC secured
	0x60 Sub-TLV:UserData is CRC secured	0x61 Sub-TLV:UserData is not CRC secured
	0x70 Sub-TLV:Time Authenticated is not CRC secured	0x70 Sub-TLV:Time Authenticated is not CRC secured

Note: The *ICV* of the Follow_Up *TLV* shall be verified, depending on RxIcvVerification. Refer to section 4.6.4.2.5.

4.6.4.2.1 SGW Calculation

[PRS_TS_00211]

Upstream requirements: RS_TS_20054

⌈If MessageCompliance is set to TRUE the SYNC_TO_GATEWAY bit within timeBaseStatus shall be set to zero.⌋

[PRS_TS_00156]

Upstream requirements: RS_TS_20053, RS_TS_20054

⌈For a Synchronized Time Base and if MessageCompliance is set to FALSE and if RxSubTLVStatus is set to TRUE the *SGW* value (Time Gateway synchronization status) shall be retrieved from the Status element of the AUTOSAR Sub-TLV:Status Secured or Sub-TLV:Status Not Secured if the *AUTOSAR TLV* in the Follow_Up message exists and if this *Sub-TLV* is part of the *AUTOSAR TLV*. If the SGW value is set to SyncToSubDomain, the SYNC_TO_GATEWAY bit within timeBaseStatus shall be set to one. Otherwise, it shall be set to zero.⌋

Note: Since a Global Time Master will not set the Time Gateway synchronization status to SYNC_TO_GATEWAY it is superfluous to transmit an *AUTOSAR Sub-TLV*:Status in this case.

[PRS_TS_00212]

Upstream requirements: RS_TS_20054

⌈If MessageCompliance is set to FALSE and if an *AUTOSAR Sub-TLV:* Status in the Follow_Up message does not exist, the SYNC_TO_GATEWAY bit within timeBaseStatus shall be set to zero.⌋

[PRS_TS_00214]

Upstream requirements: RS_TS_20061

⌈If MessageCompliance is set to FALSE and if RxSubTLVStatus is set to TRUE: if either the *AUTOSAR TLV* in the Follow_Up message does not exist or if the AUTOSAR Sub-TLV:Status Secured or Sub-TLV:Status Not Secured is not part of the *AUTOSAR TLV* a Time Slave shall discard the received Follow_Up message⌋

4.6.4.2.2 UserData Processing

[PRS_TS_00217]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈If MessageCompliance is set to FALSE and if RxSubTLVUserData is set to TRUE: if either the *AUTOSAR TLV* in the Follow_Up message does not exist or if the *AUTOSAR* Sub-TLV:UserData Secured or Sub-TLV:UserData Not Secured is not part of the *AUTOSAR TLV* a Time Slave shall discard the received Follow_Up message.⌋

[PRS_TS_00218]

Upstream requirements: RS_TS_20062

⌈If MessageCompliance is either set to TRUE or if RxSubTLVUserData is set to FALSE, a Time Slave shall discard User Data.⌋

4.6.4.2.3 CRC Validation

[PRS_TS_00267] CRC Calculation of Time Slave

Upstream requirements: RS_TS_20061

⌈The CRC calculation of the Time Slave shall use the generator polynomial *0x2F* the initial value *0xFF* andthe *XOR* value *0xFF*. Neither the input data nor the result data shall be reflected.⌋

Note: The CRC calculation is based on the AUTOSAR E2E Profile 2. For details refer to [5, FO-PRS-E2EProtocol].

[PRS_TS_00112]

Upstream requirements: RS_TS_20061

⌈The DataID shall be calculated as: DataID = DataIDList[Follow_Up.sequenceId mod 16], where DataIDList is given by configuration for the Follow_Up.⌋

Note: A specific DataID out of a predefined DataIDList ensures the identification of data elements of Time Synchronization messages.

[PRS_TS_00183]

Upstream requirements: RS_TS_20061

⌈If applying the CRC calculation on multibyte values, the byte order shall be such that the byte containing the most significant bit of the value shall be used first.⌋

[PRS_TS_00185]

Upstream requirements: RS_TS_20061

⌈If applying the CRC calculation on multibyte message data, the byte order shall be in ascending order of the octets, i.e., the octet with the lowest offset shall be used first.⌋

4.6.4.2.3.1 AUTOSAR Sub-TLV:Time Secured

[PRS_TS_00215]

Upstream requirements: RS_TS_20061

⌈If MessageCompliance is set to FALSE and if RxSubTLVTime is set to TRUE: if either the *AUTOSAR TLV* in the Follow_Up message does not exist or if the *AUTOSAR* Sub-TLV:Time Secured is not part of the *AUTOSAR TLV* a Time Slave shall discard the received Follow_Up message.⌋

[PRS_TS_00157]

Upstream requirements: RS_TS_20061

⌈If RxSubTLVTime is set to TRUE and if RxCrcValidated is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the CRC as defined in CrcFlagsRxValidated acc. to the following rule.⌋

	Validate if CrcFlagsRxValidated element is set to TRUE:	
Element	Follow_Up Message Header	Follow_Up Message Field
CrcMessageLength	messageLength	n.a.
CrcDomainNumber	domainNumber	n.a.
CrcCorrectionField	correctionField	n.a.
CrcSourcePortIdentity	sourcePortIdentity	n.a.
CrcSequenceId	sequenceId	n.a.
CrcPreciseOriginTimestamp	n.a.	preciseOriginTimestamp

[PRS_TS_00113]

Upstream requirements: RS_TS_20061

⌈If RxSubTLVTime is set to TRUE and if RxCrcValidated is set to CRC_VALIDATED or CRC_OPTIONAL, , the Time Slave shall validate the CRC for CRC_Time_0 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in CrcFlagsRxValidated acc. to the rule in the table below and the DataID. The data elements used for the calculation and thus validation of the CRC shall apply the following order:

1. the value of CRC_Time_Flags
2. the domainNumber inside the Follow_Up Message Header, if CrcDomainNumber is set to TRUE
3. the preciseOriginTimestamp inside the Follow_Up Message Field, if CrcPreciseOriginTimestamp is set to TRUE

4. the sourcePortIdentity inside the Follow_Up Message Header, if CrcSourcePortIdentity is set to TRUE
5. the DataID (refer to [PRS_TS_00112])

⌋

	For CRC_Time_0 verification required contents:	
If CrcFlagsRxValidated element is set to TRUE:	Follow_Up Message Header	Follow_Up Message Field
CrcMessageLength	n.a.	n.a.
CrcDomainNumber	domainNumber	n.a.
CrcCorrectionField	n.a.	n.a.
CrcSourcePortIdentity	sourcePortIdentity	n.a.
CrcSequenceId	n.a.	n.a.
CrcPreciseOriginTimestamp	n.a.	preciseOriginTimestamp

[PRS_TS_00114]

Upstream requirements: RS_TS_20061

⌈If RxSubTLVTime is set to TRUE and if RxCrcValidated is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the CRC for CRC_Time_1 by considering the contents of CRC_Time_Flags itself, the contents of the dependent fields as defined in CrcFlagsRxValidated acc. to the rule in the table below and the DataID. The data elements used for the calculation and thus validation of the CRC shall apply the following order:

1. the value of CRC_Time_Flags
2. the messageLength inside the Follow_Up Message Header, if CrcMessageLength is set to TRUE
3. the CrcCorrectionField inside the Follow_Up Message Header, if CrcCorrectionField is set to TRUE
4. the sequenceId inside the Follow_Up Message Field, if CrcSequenceId is set to TRUE
5. the DataID (refer to [PRS_TS_00112])

⌋

	For CRC_Time_1 verification required contents:	
If CrcFlagsRxValidated element is set to TRUE:	Follow_Up Message Header	Follow_Up Message Field
CrcMessageLength	messageLength	n.a.
CrcDomainNumber	n.a.	n.a.
CrcCorrectionField	correctionField	n.a.
CrcSourcePortIdentity	n.a.	n.a.
CrcSequenceId	sequenceId	n.a.
CrcPreciseOrigin-Timestamp	n.a.	n.a.

4.6.4.2.3.2 AUTOSAR Sub-TLV:Status secured

[PRS_TS_00115]

Upstream requirements: RS_TS_20061

⌈If RxCrcValidated is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the CRC for CRC_Status by considering the contents of Status and DataID (in this order).⌋

4.6.4.2.3.3 AUTOSAR Sub-TLV:UserData secured

[PRS_TS_00116]

Upstream requirements: RS_TS_20061

⌈If RxCrcValidated is set to CRC_VALIDATED or CRC_OPTIONAL, the Time Slave shall validate the CRC for CRC_UserData by considering the contents of UserDataLength, UserByte_0, UserByte_1, UserByte_2 and DataID (in this order).⌋

4.6.4.2.4 Sequence Counter (sequenceId) Validation

4.6.4.2.4.1 Sequence Counter Validation of SYNC Messages

Figure 4.2 illustrates the Sequence Counter validation of a Time Slave for SYNC messages.

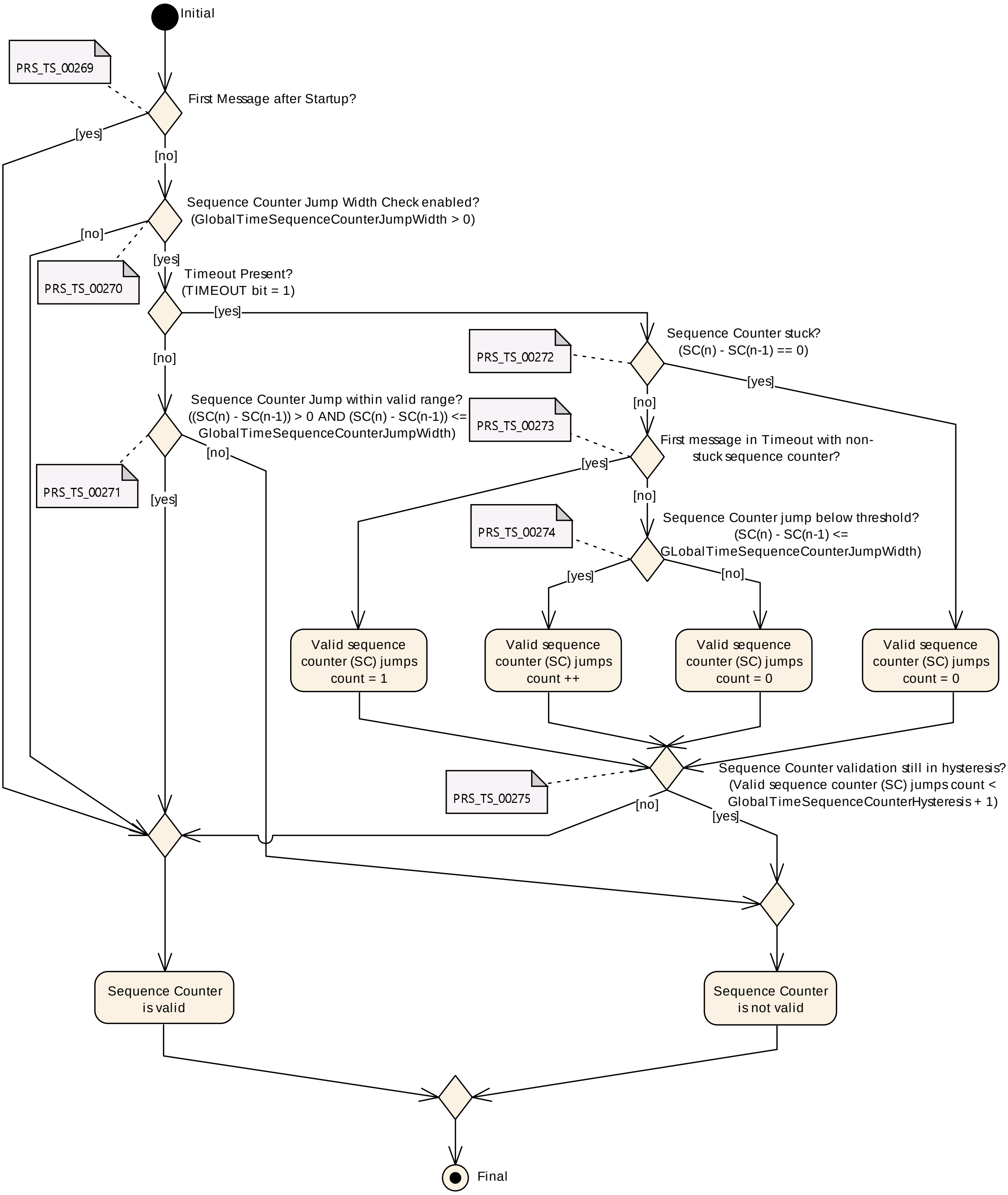
Initial
PRS_TS_00269
First Message after Startup?
[yes]
[no]
Sequence Counter Jump Width Check enabled?
(GlobalTimeSequenceCounterJumpWidth > 0)
[no]
[yes]
PRS_TS_00270
Timeout Present?
(TIMEOUT bit = 1)
[yes]
Sequence Counter stuck?
(SC(n) - SC(n-1) == 0)
PRS_TS_00272
[yes]
[no]
Sequence Counter Jump within valid range?
((SC(n) - SC(n-1)) > 0 AND (SC(n) - SC(n-1)) <=
GlobalTimeSequenceCounterJumpWidth)
PRS_TS_00273
[no]
First message in Timeout with non-stuck sequence counter?
[yes]
PRS_TS_00271
[yes]
[no]
Sequence Counter jump below threshold?
(SC(n) - SC(n-1) <=
GLobalTimeSequenceCounterJumpWidth)
PRS_TS_00274
[yes]
[no]
Valid sequence counter (SC) jumps count = 1
Valid sequence counter (SC) jumps count ++
Valid sequence counter (SC) jumps count = 0
Valid sequence counter (SC) jumps count = 0
Sequence Counter validation still in hysteresis?
(Valid sequence counter (SC) jumps count <
GlobalTimeSequenceCounterHysteresis + 1)
PRS_TS_00275
[no]
[yes]
Sequence Counter is valid
Sequence Counter is not valid
Final

Figure 4.2: Sequence Counter validation of a Time Slave for SYNC messages

[PRS_TS_00269] Sequence Counter Validation after First Sync message

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if the message is the first Sync message after startup, a Time Slave shall consider the Sequence Counter validation as successful.⌋

[PRS_TS_00270] Sequence Counter Validation if disabled

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if the Sequence Counter check is disabled (i.e., `GlobalTimeSequenceCounterJumpWidth` == 0), a Time Slave shall consider the Sequence Counter validation as successful.⌋

[PRS_TS_00271] Sequence Counter Validation if Sync Message is not first after startup

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if

- the message is not the first Sync message after startup
- and Sequence Counter check is enabled (i.e., `GlobalTimeSequenceCounterJumpWidth` > 0)
- and the referenced Time Base is not in timeout (i.e., TIMEOUT bit not set in Time Base synchronization status `timeBaseStatus`)

a Time Slave shall check the difference between the Sequence Counter of the current Sync message and the Sequence Counter of the previous Sync message.

If the difference is greater than 0 and less or equal than `GlobalTimeSequenceCounterJumpWidth`, a Time Slave shall consider the Sequence Counter validation as successful, else as failed.⌋

4.6.4.2.4.2 Sequence Counter Validation of SYNC Messages while in Timeout

This chapter specifies how to validate the Sequence Counter of SYNC messages while the Time Domain is in Timeout. When doing the validation in Timeout, the Time Slave may optionally apply a hysteresis (`GlobalTimeSequenceCounterHysteresis`, refer to [PRS_TS_00275]) to check if the Sequence Counter validation is actually successful.

This requires that a number of consecutive *Sequence Counter jumps* are valid. Requirements [PRS_TS_00272], [PRS_TS_00273] and [PRS_TS_00274] specify when an individual *Sequence Counter jump* is considered to be valid.

The optional hysteresis as part of the Sequence Counter validation improves robustness in a scenario with a buggy Time Master implementation or injection of invalid messages (Sequence Counter increments by more than GlobalTimeSequenceCounterJumpWidth). In such a scenario (without any hysteresis; refer to [PRS_TS_00273]) a Sync message with any (also invalid) Sequence Counter value would cause the Time Slave to leave the Timeout state although the Sequence Counter is not incremented correctly. A hysteresis avoids this.

[PRS_TS_00272] Sequence Counter Jump Check if Counter stuck in TIMEOUT

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if

- Sequence counter check is enabled (i.e., GlobalTimeSequenceCounterJumpWidth > 0)
- and the referenced Time Base is in timeout (i.e., TIMEOUT bit set in Time Base synchronization status timeBaseStatus)
- and Sequence Counter is stuck

a Time Slave shall consider Sequence Counter jump as invalid.⌋

Note: The Sequence Counter is considered as "stuck" (see e.g. [PRS_TS_00272], [PRS_TS_00273] and [PRS_TS_00274]), if the difference between the Sequence Counter of the current Sync message and the Sequence Counter of the previous Sync message is 0.

[PRS_TS_00273] Sequence Counter Jump Check if first Sync Message in TIMEOUT

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if

- Sequence Counter check is enabled (i.e., GlobalTimeSequenceCounterJumpWidth > 0)
- and the referenced Time Base is in timeout (i.e., TIMEOUT bit set in Time Base synchronization status timeBaseStatus)
- and the message is the first Sync message in timeout for which the Sequence Counter is not stuck,

a Time Slave shall consider the Sequence Counter jump as valid.⌋

Rationale: After a Timeout (e.g. due to a reset or disconnect of the Time Master) it is very likely that the sequence counter of the first received Sync message is out of sync, i.e., the sequence counter difference exceeds GlobalTimeSequenceCounterJumpWidth. To allow for faster re-synchronization of the sequence counter to the Time Master, the sequence counter of the first Sync message is not checked for GlobalTimeSequenceCounterJumpWidth. However, a stuck Sequence Counter will always, i.e., also in this situation, be considered as invalid (refer to [PRS_TS_00272]).

[PRS_TS_00274] Sequence Counter Jump Check if Sync Message in TIMEOUT

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if

- Sequence counter check is enabled (i.e., GlobalTimeSequenceCounterJumpWidth > 0)
- and the referenced Time Base is in timeout (i.e., TIMEOUT bit set in Time Base synchronization status timeBaseStatus)
- and the Sequence Counter is not stuck
- and the message is not the first Sync message in timeout for which the Sequence Counter is not stuck

a Time Slave shall check if the difference between the Sequence Counter of the current Sync message and the Sequence Counter of the previous Sync message exceeds the threshold GlobalTimeSequenceCounterJumpWidth.

If the difference exceeds the threshold GlobalTimeSequenceCounterJumpWidth, a Time Slave shall consider Sequence Counter jump as invalid, else as valid.⌋

[PRS_TS_00275] Sequence Counter Hysteresis Check

Upstream requirements: RS_TS_20061

⌈Upon reception of a Sync message, if

- Sequence counter check is enabled (i.e., GlobalTimeSequenceCounterJumpWidth > 0)
- and the referenced Time Base is in timeout (i.e., TIMEOUT bit set in Time Base synchronization status timeBaseStatus)

a Time Slave shall check the number of consecutive valid Sequence Counter jumps (refer to requirements [PRS_TS_00272], [PRS_TS_00273] and [PRS_TS_00274]).

If the number of consecutive valid Sequence Counter jumps exceeds the value GlobalTimeSequenceCounterHysteresis, a Time Slave shall consider the Sequence Counter validation as successful, else as failed.⌋

4.6.4.2.4.3 Sequence Counter Validation of other Messages Types

[PRS_TS_00197]

Upstream requirements: RS_TS_20061

⌈If no Follow_Up message with a matching Sequence Counter (sequenceId) and Time Domain (domainNumber) has been received within the timeout interval GlobalTimeFollowUpTimeout, the Time Slave shall discard the contents of the already received Sync message.⌋

[PRS_TS_00192]

Upstream requirements: RS_TS_20061

⌈If the Sequence Counter (sequenceId) of a received Pdelay_Resp message does not match the Sequence Counter (sequenceId) of the corresponding Pdelay_Req message, the Peer Delay Initiator shall ignore the Pdelay_Resp message.⌋

[PRS_TS_00193]

Upstream requirements: RS_TS_20061

⌈The Peer Delay Initiator shall ignore a Pdelay_Resp message, if the Pdelay_Resp message has not been received within the timeout interval GlobalTimePdelayRespAndRespFollowUpTimeout.⌋

[PRS_TS_00194]

Upstream requirements: RS_TS_20061

⌈If the Sequence Counter (sequenceId) of a received Pdelay_Resp_Follow_Up message does not match the Sequence Counter (sequenceId) of the transmitted Pdelay_Req message, the Peer Delay Initiator shall ignore the received Pdelay_Resp_Follow_Up message.⌋

[PRS_TS_00195]

Upstream requirements: RS_TS_20061

⌈The Peer Delay Initiator shall discard the content of a Pdelay_Resp message, if no Pdelay_Resp_Follow_Up message with a matching Sequence Counter (sequenceId) has been received within the timeout interval GlobalTimePdelayRespAndRespFollowUpTimeout.⌋

[PRS_TS_00196]

Upstream requirements: RS_TS_20061

⌈If the Sequence Counter (sequenceId) of a received Follow_Up message does not match the Sequence Counter (sequenceId) of the previously received Sync message of the same Time Domain (domainNumber), the Time Slave shall ignore the Follow_Up message.⌋

4.6.4.2.5 ICV Verification

[PRS_TS_00243]

Upstream requirements: RS_TS_20072

⌈If RxIcvVerification is set to ICV_IGNORED, the Time Slave shall not perform the ICV verification. If the received Follow_Up message contains the AUTOSAR *Sub-TLV*:Time Authenticated, then the Time Slave shall ignore it.⌋

[PRS_TS_00244]

Upstream requirements: RS_TS_20072

⌈If RxIcvVerification is set to ICV_OPTIONAL, the Time Slave shall not perform the ICV verification, when the received Follow_Up message does not contain the AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00245]

Upstream requirements: RS_TS_20072

⌈If RxIcvVerification is set to ICV_OPTIONAL, the Time Slave shall perform the ICV verification, when the received Follow_Up message contains the AUTOSAR *Sub-TLV*:Time Authenticated.⌋

[PRS_TS_00246]

Upstream requirements: RS_TS_20072

⌈If RxIcvVerification is set to ICV_VERIFIED, the Time Slave shall perform the ICV verification. If the received Follow_Up message does not contain the AUTOSAR *Sub-TLV*:Time Authenticated, then the ICV verification shall be assessed as unsuccessful.⌋

[PRS_TS_00247]

Upstream requirements: RS_TS_20072

⌈If RxIcvVerification is set to ICV_NOT_VERIFIED, the Time Slave shall not perform the ICV verification and the received Follow_Up message shall not contain the AUTOSAR *Sub-TLV*:Time Authenticated. If the received Follow_Up message contains the AUTOSAR *Sub-TLV*:Time Authenticated, then the Time Slave shall not perform the ICV verification and ICV verification shall be assessed as unsuccessful.⌋

[PRS_TS_00248]

Upstream requirements: RS_TS_20072

⌈As initial step of ICV verification process, if FVL is greater than 0 and 'ICV with FV' bit is set in ICV_Flags of the received `Follow_Up` message, then the Time Slave shall derive the FV and perform the FV verification. If the Time Slave fails to derive the FV and FV verification is unsuccessful, then the ICV verification is unsuccessful.⌋

[PRS_TS_00249]

Upstream requirements: RS_TS_00039, RS_TS_20072

⌈During the ICV verification process if 'ICV with FV' bit is not set in ICV_Flags of received `Follow_Up` message, the Time Slave shall not include the FV in the ICV verification.⌋

[PRS_TS_00250]

Upstream requirements: RS_TS_00039, RS_TS_20072

⌈During the ICV verification process if FVL is equal to 0 and 'ICV with FV' bit is set in ICV_Flags of received `Follow_Up` message, the Time Slave shall not derive the FV and the ICV verification is unsuccessful.⌋

[PRS_TS_00251]

Upstream requirements: RS_TS_20072

⌈When the received `Follow_Up` message contains multiple AUTOSAR *Sub-TLV*:Time Authenticated, the Time Slave shall aggregate the ICV value correctly (refer to [PRS_TS_00227], [PRS_TS_00228], [PRS_TS_00229], [PRS_TS_00230], [PRS_TS_00231]). If the Time Slave cannot aggregate the ICV value correctly (e.g., incorrect sequence numbers, length), then ICV verification is unsuccessful.⌋

[PRS_TS_00252]

Upstream requirements: RS_TS_20072

⌈If the ICV verification (Inclusive of FV verification time) takes longer than the timeout `IcvVerificationTimeout`, then ICV verification is unsuccessful.⌋

[PRS_TS_00258]

Status: DRAFT

Upstream requirements: RS_TS_20072

⌈During the ICV verification process, if the 'ICV generation failed' bit is set in ICV_Flags, the ICV verification is considered unsuccessful.⌋

4.6.4.2.6 Message Disassembling

[PRS_TS_00118]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈If the Type of a *Sub-TLV* cannot be recognized at the receiver side, it shall be ignored and the next subsequent *Sub-TLV* shall be evaluated.⌋

Note: The Length field of each *Sub-TLV* is always at the same position within each *Sub-TLV*. It will be used to jump over the unknown *Sub-TLV* to the next Type field.

[PRS_TS_00119]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈If any of the following conditions is not met, a Time Slave shall consider the validation of received `Sync` or `Follow_Up` message is not successful, discard a received `Sync` or `Follow_Up` message and reset the sequence (ie., waiting for next `Sync` message):

1. Sequence Counter (`sequenceId`) is valid (refer to [PRS_TS_00269], [PRS_TS_00270], [PRS_TS_00271], [PRS_TS_00272], [PRS_TS_00273], [PRS_TS_00274], [PRS_TS_00275]).
2. If `Follow_Up`: `Follow_Up.TLV[AUTOSAR].Sub-TLV.Type` matches depending on configuration of `RxCrcValidated`
3. The Time Domain matches to one of the configured Time Domains
4. If `Follow_Up`: All *CRCs* are successfully validated depending on the configuration of `RxCrcValidated` and `CrcFlagsRxValidated`.
5. If `Follow_Up`: The `Length` field for every "known", i.e., *Sub-TLV* that is contained in the AUTOSAR TLV matches the specified value for this *Sub-TLV*.
6. If `Follow_Up`: The AUTOSAR TLV Header's `lengthField` is equal to the accumulated length of all Sub-TLVs plus `6`.
7. If `Follow_Up`: The *ICV* is successfully verified depending on the configuration of `RxIcvVerification`.
8. The nanoseconds element of the `preciseOriginTimestamp` matches the range of [0 .. 999999999] ns.

⌋

Note: Section 4.6.3.2.4 provides more details on the Length field of every *Sub-TLV*.

[PRS_TS_00120]

Upstream requirements: RS_TS_20061, RS_TS_20062

⌈For each received Time Synchronization message, the Time synchronization protocol shall disassemble the message after successful validation.⌋

4.7 Time measurement with Switches

In a time aware Ethernet network, two basic HW types of control units exists:

1. Endpoints directly working on a local Ethernet-Controller
2. Time Gateways, resp. Time Aware Bridges, where the local Ethernet-Controller connects to an external Switch device.

The extension "Time measurement with Switches" focusses on 2. A Switch device leads to additional delays, which have to be considered for the calculation of the corresponding Time Base. Additionally, the support of time stamping in HW is a Switch-Port specific feature, which leads to an extension of the used function APIs. These APIs enabling a Switch port specific detection of ingress and egress messages together with a given timestamp, if enabled.

If the Switch Management and Global Time support is implemented as a part of the program running on the Switch HW, this will not be considered by 2. For this case, the behavior can be seen as described in 1.

[PRS_TS_00053]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈Time measurement with Switches supports the use case "Time Aware Bridge with GTM as Management CPU" like shown in Figure 4.3.⌋

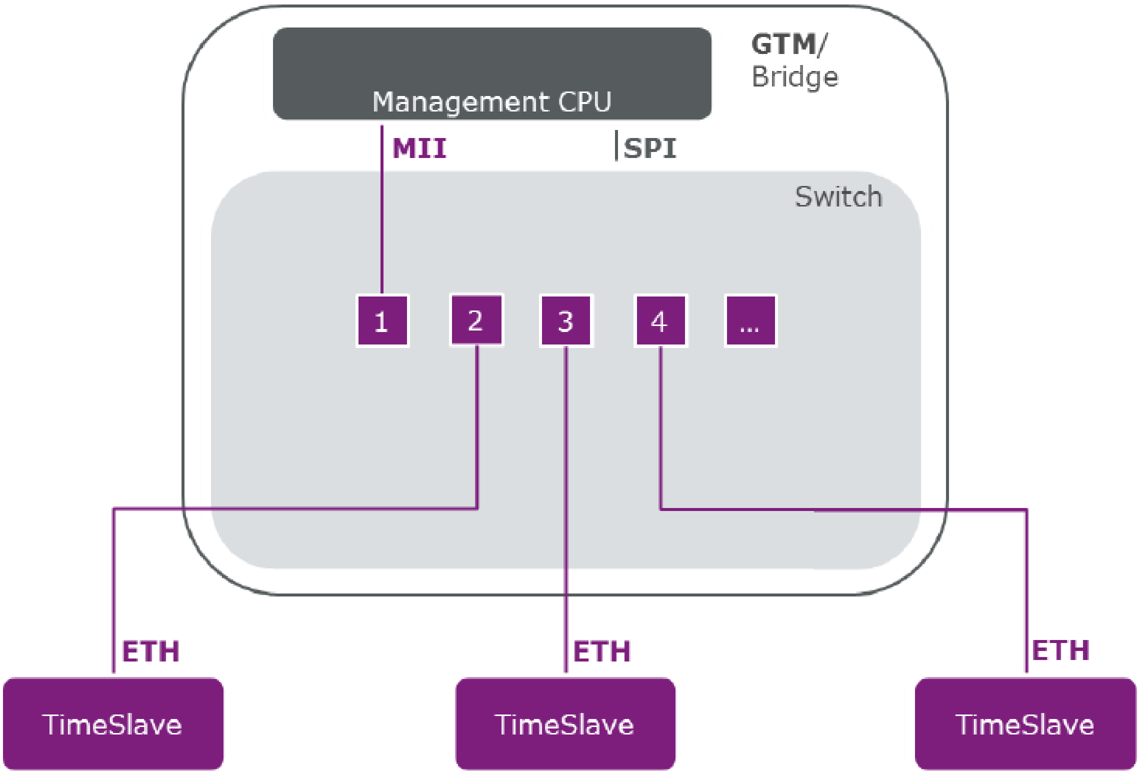

Figure 4.3: Time Aware Bridge with GTM as Management CPU

[PRS_TS_00054]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈Time measurement with Switches supports the use case "Time Aware Bridge with GTM not as Management CPU" like shown in Figure 4.4.⌋

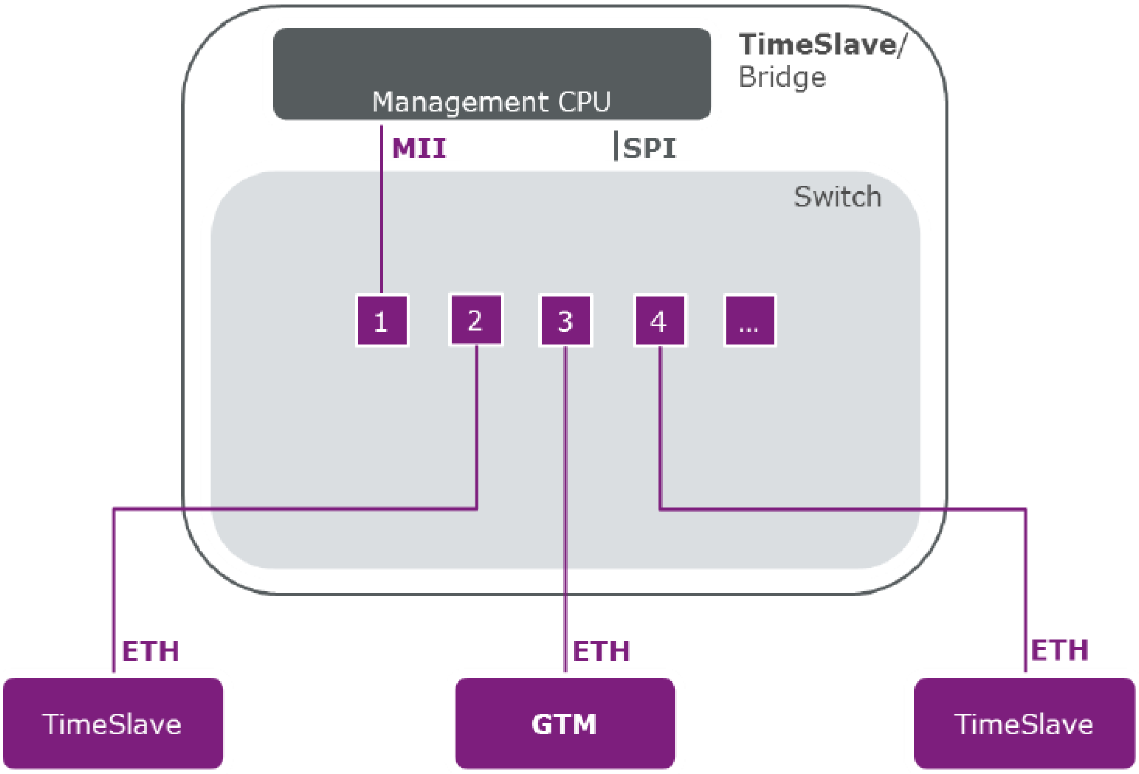

Figure 4.4: Time Aware Bridge with GTM not as Management CPU

4.8 Pdelay and Time Synchronization measurement point

[PRS_TS_00055]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈The path delay measurement will be done always as Port-to-Port measurement like specified in in [1, IEEE 802.1 AS] chapter 11.1.2 Propagation delay measurement for the device external Ethernet path.⌋

[PRS_TS_00056]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈The inner delay of the Ethernet path (Residence Time) is determined at the time where `Sync` is received and transmitted, by using the message specific ingress and egress timestamps.⌋

Note: This belongs to the fact, that the Residence Time might be discontinuous, depending on the current busload, while `Sync` messages are transmitted / received, the Switch HW architecture and the message forwarding method. A static delay measurement method for this part of the communication path might lead to an unprecise time measurement. Nevertheless, static Residence Time parameters are considered by this specification, to increase the performance while calculating the Global Time resp. the `correctionField` and the flexibility to support different Switch devices, such as Switches, which do not support time stamping on each ingress or egress port.

4.9 Time Aware Bridge with GTM as Management CPU

[PRS_TS_00057]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈Time measurement with Switches supporting the use case "Time Aware Bridge with GTM as Management CPU" following the given timestamping points like shown in Figure 4.5 and Figure 4.6.⌋

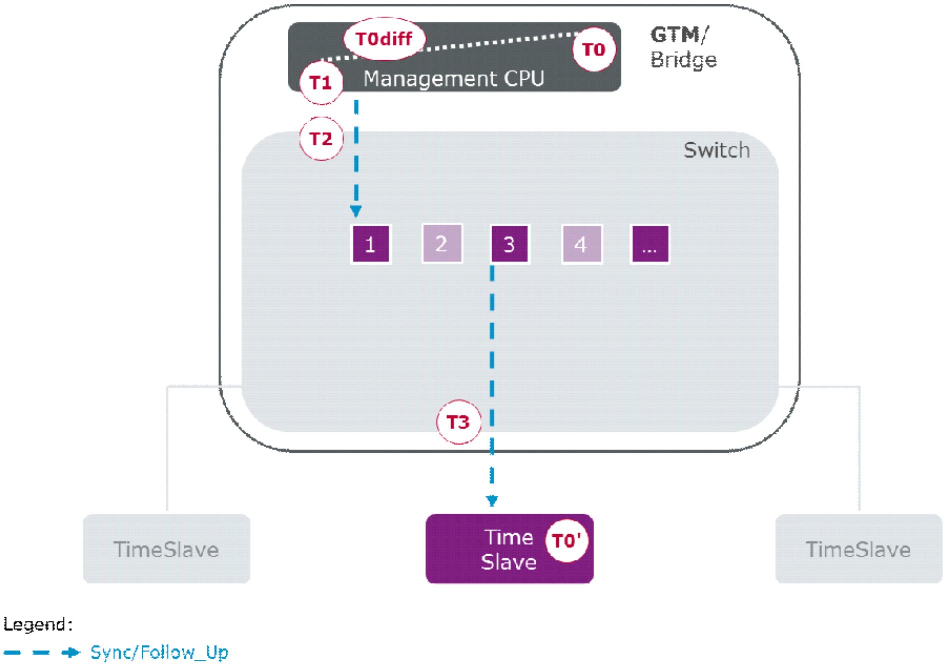

Figure 4.5: Sync/Follow_Up message flow with Timestamping points for Sync for Time Aware Bridge with GTM as Management CPU

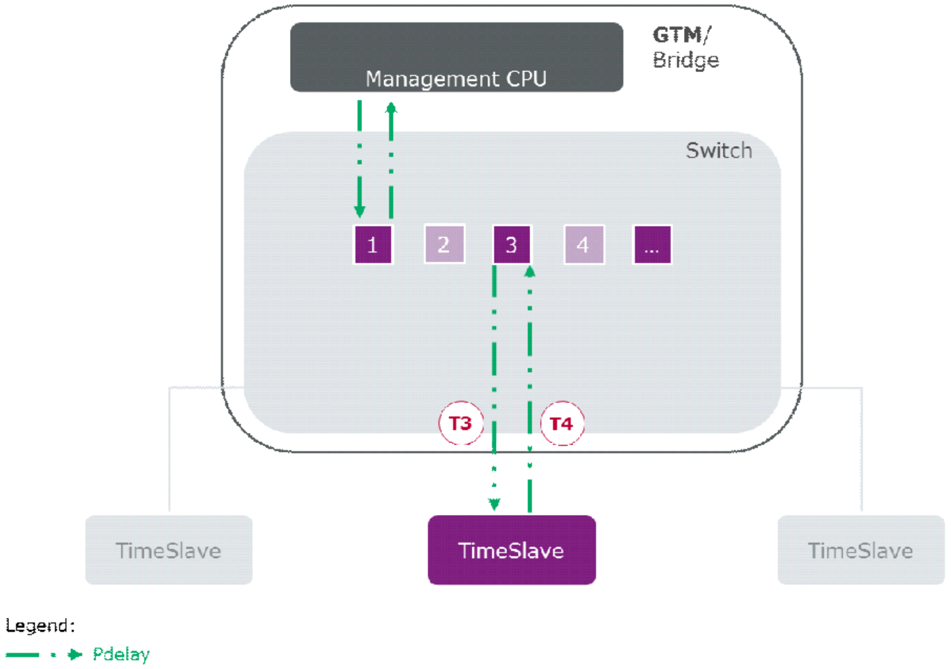

Figure 4.6: Pdelay message flow with Timestamping points for Time Aware Bridge with GTM as Management CPU

Note: The picture Figure 4.5 and Figure Figure 4.6 shows an example Port selection as simplification.

[PRS_TS_00058]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈Time measurement with Switches supporting the use case "Time Aware Bridge with GTM as Management CPU" considers the inner Switch delay by a modification of the `correctionField` as well as `Pdelay` timestamping for `requestReceiptTimestamp` and `responseOriginTimestamp` like shown in Figure 4.7.⌋

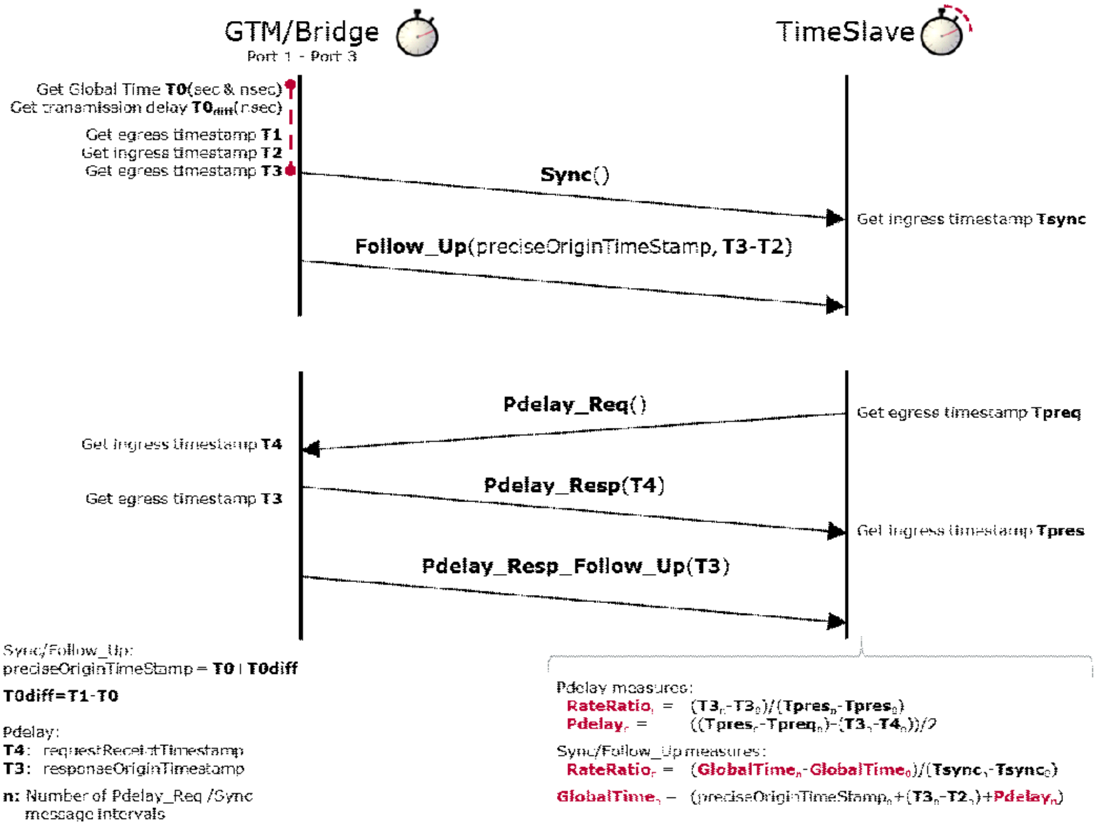
GTM/Bridge
Port 1 - Port 3
TimeSlave
Get Global Time T0(sec & nsec)
Get transmission delay T0diff(nsec)
Get egress timestamp T1
Get ingress timestamp T2
Get egress timestamp T3
Sync()
Get ingress timestamp Tsync
Follow_Up(preciseOriginTimeStamp, T3-T2)
Pdelay_Req()
Get egress timestamp Tpreq
Get ingress timestamp T4
Pdelay_Resp(T4)
Get egress timestamp T3
Get ingress timestamp Tpres
Pdelay_Resp_Follow_Up(T3)
Sync/Follow_Up:
preciseOriginTimeStamp = T0 + T0diff
T0diff=T1-T0
Pdelay:
T4: requestReceiptTimestamp
T3: responseOriginTimestamp
n: Number of Pdelay_Req /Sync message intervals
Pdelay measures:
RateRatio_n = (T3_n-T3_0)/(Tpres_n-Tpres_0)
Pdelay_n = ((Tpres_n-Tpreq_n)-(T3_n-T4_n))/2
Sync/Follow_Up measures:
RateRatio_n = (GlobalTime_n-GlobalTime_0)/(Tsync_n-Tsync_0)
GlobalTime_n = (preciseOriginTimeStamp_n+(T3_n-T2_n)+Pdelay_n)

Figure 4.7: Timestamping sequence for Time Aware Bridge with GTM as Management CPU

Note: The calculation in Figure 4.7 shows an example Port selection as simplification.

[PRS_TS_00166]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If GlobalTimeUplinkToTxSwitchResidenceTime is set to 0, the Ethernet module shall ignore this parameter and measure the inner delay of the Switch egress Ethernet path (Uplink to Tx Residence Time (**T3 - T2**)) by using always the ingress (**T2**) and egress (**T3**) timestamp as given in Figure 4.7.⌋

[PRS_TS_00167]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If GlobalTimeUplinkToTxSwitchResidenceTime is greater than 0, the Ethernet module shall use this parameter as value for the inner delay of the Switch egress Ethernet path (Uplink to Tx Residence Time (**T3 - T2**)) instead of using the measurement method described in [PRS_TS_00166].⌋

4.10 Time Aware Bridge with GTM not as Management CPU

[PRS_TS_00059]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈Time measurement with Switches supporting the use case Time Aware Bridge with GTM not as Management CPU following the given timestamping points like shown in Figure 4.8 and Figure 4.9.⌋

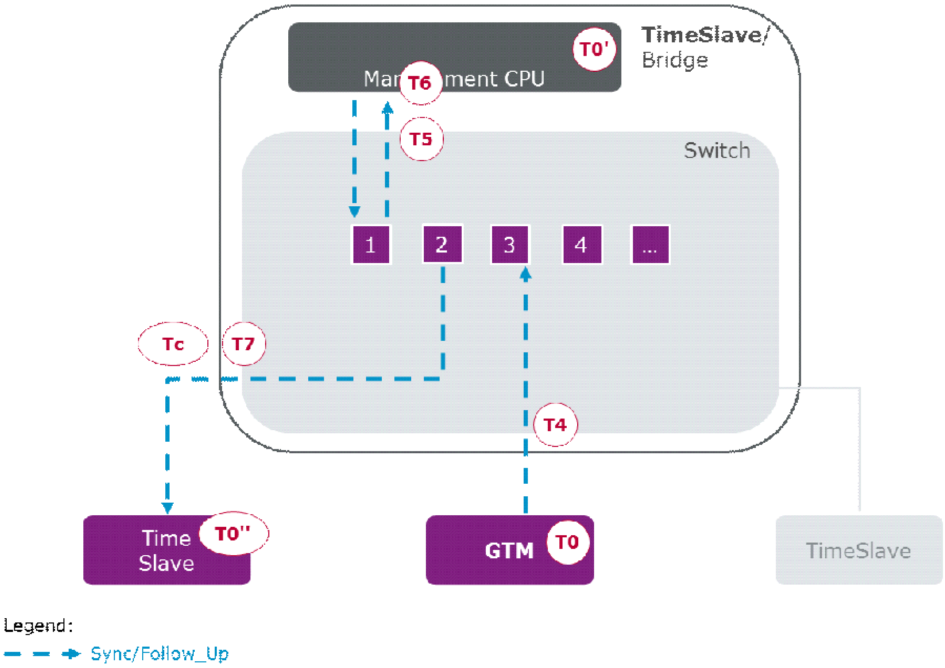

Figure 4.8: Sync/Follow_Up message flow with Timestamping points for Sync for Time Aware Bridge with GTM not as Management CPU

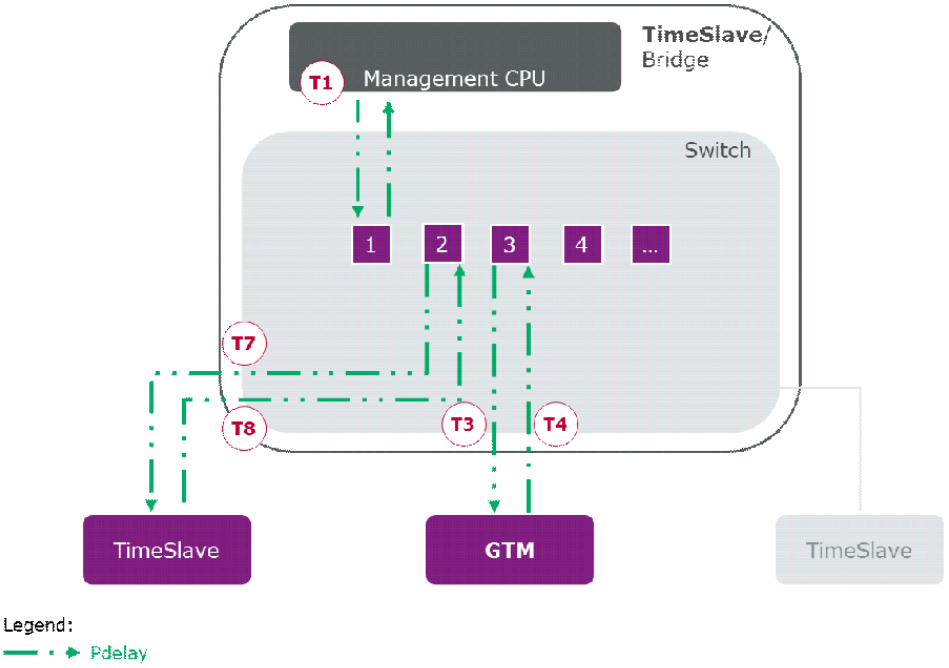

Figure 4.9: Pdelay message flow with Timestamping points for Time Aware Bridge with GTM not as Management CPU

[PRS_TS_00060]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈Time measurement with Switches supporting the use case Time Aware Bridge with GTM not as Management CPU considers the inner Switch delay by a modification of the `correctionField` as well as `Pdelay` timestamping for `requestReceiptTimestamp` and `responseOriginTimestamp`.⌋

[PRS_TS_00207]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If the `Follow_Up` message contains an AUTOSAR *TLV*, which contains a *Sub-TLV*:Time Secured it shall be checked, if the element `CRC_Time_Flags` contains BitMask `0x04` (i.e., the content of `correctionField` is `CRC` protected). If this bit is set then the validation of the `CRC_Time_1` element shall be done. The data elements used for the calculation and thus validation of the CRC shall be applied with the following order:

1. the value of `CRC_Time_Flags`
2. the `length of the message` inside the `Follow_Up` Message Header, if the element `CRC_Time_Flags` contains BitMask `0x01`
3. the `correctionField` inside the `Follow_Up` Message Header
4. the `sequenceId` inside the `Follow_Up` Message Header, if the element `CRC_Time_Flags` contains BitMask `0x10`

5. the *DataID*

⌋

Note: The *CRC* Validation shall be done as specified in section 4.6.4.2.3.

[PRS_TS_00208]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If the CRC validation of an *AUTOSAR TLV* fails, the Follow_Up message shall be dropped instead of being forwarded.⌋

[PRS_TS_00209]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If the CRC validation of an *AUTOSAR TLV* is successful, the correctionField shall be modified and the element CRC_Time_1 inside the Sub-TLV:Time Secured shall be calculated according to the content of the CRC_Time_Flags element.⌋

[PRS_TS_00253]

Upstream requirements: RS_TS_20072

⌈If the Follow_Up message contains an AUTOSAR *TLV*, which contains AUTOSAR *Sub-TLV*:Time Authenticated, then the Time Aware Bridge shall verify the ICV.⌋

[PRS_TS_00254]

Upstream requirements: RS_TS_20072

⌈If the *ICV* verification of the Follow_Up message fails, then the Follow_Up message shall be dropped instead of being forwarded.⌋

[PRS_TS_00255]

Upstream requirements: RS_TS_20072

⌈If the *ICV* verification of the Follow_Up message is successful, then the following shall be done:

1. CrcCorrectionField shall be modified inside the *Sub-TLV*:Time Secured
2. the new AUTOSAR *Sub-TLV*:Time Authenticated is constructed for the updated Follow_Up
3. the old AUTOSAR *Sub-TLV*:Time Authenticated is replaced with the new AUTOSAR *Sub-TLV*:Time Authenticated in the Follow_Up message

⌋

[PRS_TS_00168]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If rx_residence_time is set to 0, the Time Synchronization over Ethernet shall ignore this parameter and measure the inner delay of the Switch ingress Ethernet path (Rx to Uplink Residence Time (T5 - T4)) by using always the ingress (T4) and egress (T5) timestamp.⌋

[PRS_TS_00171]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If rx_residence_time is greater than 0, the Time Synchronization over Ethernet shall use this parameter as value for the inner delay of the Switch ingress Ethernet path (Rx to Uplink Residence Time (T5 - T4)) instead of using the measurement method.⌋

[PRS_TS_00169]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If rx_residence_time and tx_residence_time are set to 0, the Ethernet module shall ignore both parameter and measure the inner delay of the Switch ingress and egress Ethernet path (Rx to Uplink and Uplink to Tx Residence Time (T7 to T4)) by using always the ingress (T4) and egress (T7) timestamp.⌋

[PRS_TS_00170]

Upstream requirements: RS_TS_20048, RS_TS_20059

⌈If rx_residence_time and tx_residence_time are greater than 0, the Ethernet module shall use the sum of both parameter for the value of the inner delay of the Switch ingress and egress Ethernet path (Rx to Uplink and Uplink to Tx Residence Time (T7 to T4)) instead of using the measurement method⌋

Note: A separate Uplink to Tx Residence Time (T7 to $T_{UplinkMmCpu}$) replacement by using tx_residence_time might be also possible, but is not considered by the scenario.

4.11 Error messages

There are no dedicated error messages defined in IEEE Standard 802.1AS-30 [1, IEEE 802.1 AS].

4.12 Security Events

Security Events handling is specified in the corresponding classic and adaptive platform documents.

5 Configuration parameters

The Following chapter summarizes all the configuration parameters that are used.

Name	Description
RateRatioEnable	This parameter enables/disables the calculation of the rate ratio based on the neighbor rate ratio.
RateRatioMeasurementCount	This parameter gives the number of successive, successful pDelay measurements used to calculate neighborRateRatio according to [1, IEEE 802.1 AS].
CRC_Support	represents whether the CRC configuration is supported or not
rx_residence_time	This parameter is specifying the default value used for the residence time
tx_residence_time	This parameter is specifying the default value used for the residence time
FramePrio	This optional parameter, if present, indicates the priority of outgoing messages, if sent via VLAN (used for the 3-bit PCP field of the VLAN tag). If this optional parameter is not present, frames are sent without a priority and VLAN field.
GlobalTimeTxPdelayReqPeriod	This parameter represents configuration of the TX period for Pdelay_Req messages. A value of 0 disables the cyclic Pdelay measurement.
PdelayLatencyThreshold	Threshold for calculated Pdelay. If a measured Pdelay exceeds PdelayLatencyThreshold, this value is discarded.
PdelayRespAndResp-FollowUpTimeout	Timeout value for Pdelay_Resp and Pdelay_Resp_Follow_Up after a Pdelay_Req has been transmitted resp. a Pdelay_Resp has been received. A value of 0 deactivates this timeout observation.
GlobalTimePropagationDelay	If cyclic propagation delay measurement is enabled, this parameter represents the default value of the propagation delay until the first actually measured propagation delay is available. If cyclic propagation delay measurement is disabled, this parameter replaces a measured propagation delay by a fixed value.
GlobalTimePdelayRespEnable	This parameter allows disabling Pdelay_Resp, Pdelay_Resp_Follow_Up transmission, if no Pdelay_Req messages are expected. FALSE: No Pdelay requests expected. Pdelay_Resp / Pdelay_Resp_Follow_Up transmission is disabled. TRUE: Pdelay requests expected. Pdelay_Resp, Pdelay_Resp_Follow_Up transmission is enabled.
GlobalTimeTxPeriod	This parameter represents configuration of the TX period.
GlobalTimeFollowUpTimeout	Timeout value of the Follow_Up message (of the subsequent Sync message). A value of 0 deactivates this timeout observation.
MasterSlaveConflictDetection	Enables master / slave conflict detection and notification. true: detection and notification is enabled. false: detection and notification is disabled.

MessageCompliance	true: IEEE 802.1AS compliant message format will be used. false: IEEE 802.1AS message format with AUTOSAR extension will be used.
RxCrcValidated	• CRC_IGNORED (ignores any CRC inside the Sub-TLVs) • CRC_NOT_VALIDATED (If MessageCompliance is set to FALSE: Ethernet discards Follow_Up messages with Sub-TLVs of Type 0x28, 0x50 or 0x60) • CRC_OPTIONAL (If MessageCompliance is set to FALSE: Ethernet discards Follow_Up messages with Sub-TLVs of Type 0x28, 0x50 or 0x60, that contain an incorrect CRC value.) • CRC_VALIDATED (If MessageCompliance is set to FALSE: Ethernet discards Follow_Up messages with Sub-TLVs of Type 0x28, 0x50 or 0x60, that contain an incorrect CRC value. Ethernet rejects Follow_Up messages with Sub-TLVs of Type 0x51 or 0x61)
CrcFlagsRxValidated	This container collects definitions which parts of the Follow_Up message elements shall be included in the CRC validation.
CrcMessageLength	messageLength from the Follow_Up Message Header shall be included in CRC calculation.
CrcDomainNumber	domainNumber from the Follow_Up Message Header shall be included in CRC calculation.
CrcCorrectionField	correctionField from the Follow_Up Message Header shall be included in CRC calculation.
CrcSourcePortIdentity	sourcePortIdentity from the Follow_Up Message Header shall be included in CRC calculation.
CrcSequenceId	sequenceId from the Follow_Up Message Header shall be included in CRC calculation.
CrcPreciseOriginTimestamp	preciseOriginTimestamp from the Follow_Up Message Field shall be included in CRC calculation.
GlobalTimeUplinkToTxSwitchResidenceTime	This parameter is specifying the default value used for the residence time of the Ethernet Switch [Uplink to Egress]. This value is used by the Ethernet module if the calculation of the residence time failed.
TxSubTLVTime	This represents the configuration whether a `Sub-TLV:Time Secured` shall be sent by the Time Master within the AUTOSAR TLV.
TxSubTLVStatus	This represents the configuration whether a `Sub-TLV:Status Secured` or `Sub-TLV:Status Not Secured` shall be sent by the Time Master within the AUTOSAR TLV.
TxSubTLVUserData	This represents the configuration whether a `Sub-TLV:UserData Secured` or `Sub-TLV:UserData Not Secured` shall be sent by the Time Master within the AUTOSAR TLV.

RxSubTLVTime	This represents the configuration whether a `Sub-TLV:Time Secured` within the AUTOSAR TLV shall be processed by the Time Slave or Time Gateway.
RxSubTLVStatus	This represents the configuration whether a `Sub-TLV:Status Secured` or `Sub-TLV:Status Not Secured` within the AUTOSAR TLV shall be processed by the Time Slave or Time Gateway.
RxSubTLVUserData	This represents the configuration whether a `Sub-TLV:UserData Secured` or `Sub-TLV:UserData Not Secured` within the AUTOSAR TLV shall be processed by the Time Slave or Time Gateway.
TLVFollowUpICVSubTLV	This represents the configuration of whether an AUTOSAR Follow_Up TLV Time Authenticated Sub-TLV is used or not.
CrcTimeFlagsTxSecured	This item collects definitions which parts of the Follow_Up message elements shall be used for CRC calculation.
GlobalTimeTxCrcSecured	This represents the configuration of whether or not CRC is supported.
GlobalTimeSequenceCounterJumpWidth	GlobalTimeSequenceCounterJumpWidth specifies the maximum allowed jump of the Sequence Counter between consecutive two Sync messages.
GlobalTimePdelayRespAndRespFollowUpTimeout	Timeout value for Pdelay_Resp and Pdelay_Resp_Follow_Up after a Pdelay_Req has been transmitted resp. a Pdelay_Resp has been received.
IcvGenerationTimeout	This represents the configuration of timeout value for the ICV calculation.
IcvVerificationTimeout	This represents the configuration of timeout value for the ICV verification.
RxIcvVerification	• ICV_IGNORED (the ICV verification of received Follow_Up messages is ignored. If AUTOSAR *Sub-TLV*:Time Authenticated is present, then ICV verification will not be performed.) • ICV_OPTIONAL (the ICV verification of received Follow_Up messages is performed when it contains the AUTOSAR *Sub-TLV*:Time Authenticated.) • ICV_VERIFIED (the ICV verification of received Follow_Up messages is performed, i.e., the received Follow_Up messages shall contain the AUTOSAR *Sub-TLV*:Time Authenticated.) • ICV_NOT_VERIFIED (the ICV verification of received Follow_Up messages is not performed, i.e., the received Follow_Up messages shall not contain the AUTOSAR *Sub-TLV*:Time Authenticated.)
tx_debounce_time	This represents the configuration of timeout value for the transmission of ptp frames.
rx_debounce_time	This represents the configuration of timeout value for not receiving the Follow_Up message after Sync is received.
GlobalTimeIcvFvLength	This represents the configuration of length of FV in the AUTOSAR Sub-TLV:Time Authenticated.

GlobalTimeIcvLength	This represents the configuration of length of ICV in the AUTOSAR Sub-TLV:Time Authenticated.
GlobalTimeIcvCryptoPrimitive	This represents the configuration of cryptographic primitive used for ICV generation and ICV verification.
GlobalTimeSequenceCounterHysteresis	GlobalTimeSequenceCounterHysteresis specifies the number of consecutive valid message pairs required by the Time Slave, when it is in a Timeout state, before it can revalidate and consider the time as valid again.

Table 5.1: Configuration Parameters

6 Protocol usage and guidelines

Please note that chapter 5 provides several requirements on usage.

7 References

[1] IEEE Std 802.1AS-2011

[2] Explanation of Time Sensitive Network features
AUTOSAR_FO_EXP_TimeSensitiveNetworkFeatures

[3] IEEE 802.1Q-2011 - IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks

[4] IEEE 1588-2019: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

[5] E2E Protocol Specification
AUTOSAR_FO_PRS_E2EProtocol

A Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

A.1 Traceable item history of this document according to AUTOSAR Release R24-11

A.1.1 Added Specification Items in R24-11

[PRS_TS_00265] [PRS_TS_00266] [PRS_TS_00267] [PRS_TS_00269] [PRS_TS_00270] [PRS_TS_00271] [PRS_TS_00272] [PRS_TS_00273] [PRS_TS_00274] [PRS_TS_00275]

A.1.2 Changed Specification Items in R24-11

[PRS_TS_00003] [PRS_TS_00011] [PRS_TS_00012] [PRS_TS_00094] [PRS_TS_00119] [PRS_TS_00140] [PRS_TS_00141] [PRS_TS_00149] [PRS_TS_00164]

A.1.3 Deleted Specification Items in R24-11

[PRS_TS_00084] [PRS_TS_00085] [PRS_TS_00086] [PRS_TS_00095] [PRS_TS_00103] [PRS_TS_00110] [PRS_TS_00117] [PRS_TS_00198] [PRS_TS_00199] [PRS_TS_00200] [PRS_TS_00213] [PRS_TS_00216]

A.2 Traceable item history of this document according to AUTOSAR Release R23-11

A.2.1 Added Specification Items in R23-11

[PRS_TS_00256] [PRS_TS_00257] [PRS_TS_00258] [PRS_TS_00259] [PRS_TS_00260] [PRS_TS_00261] [PRS_TS_00262] [PRS_TS_00263] [PRS_TS_00264]

A.2.2 Changed Specification Items in R23-11

[PRS_TS_00003] [PRS_TS_00070] [PRS_TS_00071] [PRS_TS_00085] [PRS_TS_00104] [PRS_TS_00119] [PRS_TS_00206] [PRS_TS_00207] [PRS_TS_00220] [PRS_TS_00238]

A.2.3 Deleted Specification Items in R23-11

none