Document Title	Specification of Time Synchronization
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	880

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Offset Time Base feature removed
2023-11-23	R23-11	AUTOSAR Release Management	• Start of transmission of sync message updated • List of Adaptive Platform Functional Clusters added • Application errors added
2022-11-24	R22-11	AUTOSAR Release Management	• Support for restorage of Global Time from persistent memory added • Several minor clarifications and corrections
2021-11-25	R21-11	AUTOSAR Release Management	• Document clean-up, Title changed • Chapter "9 Sequence diagrams" temporarily deleted for clean-up purposes • Uptraces updated • Review findings (terminology, typos, etc.) resolved

△

2020-11-30	R20-11	AUTOSAR Release Management	• TSYNC API redesign and requirments updates • Harmomized with CP and RS Documents • Document adapted to new template • Terminology clarification and cleanup
2019-11-28	R19-11	AUTOSAR Release Management	• Requirements traceability changed to Foundation RS TimeSync specification • Add Time Validation • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	• Functional description detached from actual API • Improved resource discovery
2018-10-31	18-10	AUTOSAR Release Management	• Minor changes and bugfixes • Editorial changes
2018-03-29	18-03	AUTOSAR Release Management	• Class design changed to ensure type safety • API related sections moved from chapter 7 to chapter 8 • Minor changes and bugfixes
2017-10-27	17-10	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

Time Synchronization between different applications and/or ECUs is of paramount importance when correlation of different events across a distributed system is needed, either to be able to track such events in time or to trigger them at an accurate point in time.

For this reason, a Time Synchronization API is offered to the Application, so it can retrieve the time information synchronized with other entities / ECUs.

For the format, message sequences and semantics of the time synchronization protocols to use, please refer to the Protocol Requirements Specicification (PRS) of the AUTOSAR Time synchronization Protocol (see [1]).

The Time Synchronization functionality is then offered by means of different "Time Base Resources" (from now on referred to as TBR).

These TBRs are classified in different types. These types have an equivalent design to the types of the time bases offered in the Synchronized Time Base Manager specification [2] (from now on referred to as StbM). The classification is the following:

- Synchronized Master Time Base
- Synchronized Slave Time Base

As in StbM, the TBRs offered by the Time Synchronization module (TS from now on), are also synchronized with other Time Bases on other nodes of a distributed system.

The Application consumes the time information provided and managed by the TBRs. Therefore, the TBRs serve as Time Base brokers, offering access to Synchronized Time Bases. By doing so, the TS module abstracts from the "real" Time Base provider.

The TS module supports securing of the Time Synchronization messages sent via network.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations that are only relevant within this specification. A general list of acronyms and abbreviations is available in [3].

2.1 Acronyms and Abbreviations

Abbreviation / Acronym:	Description:
backupTimestamp	Value of the Global Time which is stored into persistent memory and used at startup for calculating the initial value of the Time Base.
Clock Update Counter	Counter value belonging to the Time Base, that indicates a Time Base update.
DST	Daylight Saving Time, also know as Day Light Saving (abbreviated DLS), is the practice of advancing clocks during summer months so that evening daylight lasts longer, while sacrificing normal sunrise times. Typically, regions that use daylight saving time adjust clocks forward one hour close to the start of spring and adjust them backward in the autumn to standard time.
FVM	Freshness Value Manager
FV	Freshness Value
gPTP	Generalized Precision Time Protocol
ICV	Integrity Check Value
NTP	Network Time Protocol
OS	Operating 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
$\text{Pdelay}_{RespFollowUp}$	Propagation / path delay Follow-Up message
PTP	Precision Time Protocol
r_{oc}	Rate for time offset elimination via Rate Adaption.
r_{rc}	Current rate for correcting the local instance of the Time Base.
StbM	Synchronized Time Base Manager
Sync	Time synchronization message (Sync)
TBR	Time Base Resource
TCorrInt	OffsetCorrectionAdaptionInterval
TG	Received value of the Global Time.
TG_{Start}	Global Time part of the Updated Rx Time Tuple taken at the start of a Rate Correction measurement.
TG_{Stop}	Global Time part of the Updated Rx Time Tuple taken at the end of a Rate Correction measurement.
Timesync	Time Synchronization (Refers to the action of Synchronizing the Time by means of a time synchronization protocol/bus/messages).
TL_{Sync}	Value of the local instance of the Time Base before the new value of the Global Time is applied.
TLV	Type, Length, Value field (acc. to IEEE 802.1AS)
TS	Time Synchronization
TSP	A bus specific Time Synchronization Provider.
TV	Current value of the Virtual Local Time.

Abbreviation / Acronym:	Description:
TV_{Start}	Virtual Local Time part of the Updated Rx Time Tuple taken at the start of a Rate Correction measurement.
TV_{Stop}	Virtual Local Time part of the Updated Rx Time Tuple taken at the end of a Rate Correction measurement.
TV_{Sync}	Value of the Virtual Local Time
UTC	Coordinated Universal Time

Table 2.1: Acronyms and Abbreviations

2.2 Definitions

2.2.1 ara::core::SteadyClock

Definition: TS is using `ara::core::SteadyClock` as the basis for its interfaces and for synchronization with the daemon process realizing the time-sync protocol.

2.2.2 Time Base Application

1. **Active Application**
   This kind of Application autonomously calls the TS either:
   - To read time information from the TBRs
   - To update the Time Base maintained by a TBR, according to application information.
2. **Notification Application**
   This feature will be provided at a later release/version of the TS.

2.2.3 Rate Correction

Definition: A factor that is applied to the Local Time to correct the Rate Deviation between the Local Clock's frequency and the Master Clock frequency. The value is typically close to 1.0.

2.2.4 Rate Deviation

Definition: Positive or negative rate offset to an rate ratio of 1.0 betweeen the Local Clock's frequency and the Master Clock's frequency, i.e., how much the frequency of the Local Clock deviates from that of the Master clock. The value should typically be close to 0.0.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Time Synchronization Protocol Specification
AUTOSAR_FO_PRS_TimeSyncProtocol

[2] Specification of Synchronized Time-Base Manager
AUTOSAR_CP_SWS_SynchronizedTimeBaseManager

[3] Glossary
AUTOSAR_FO_TR_Glossary

[4] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core

[5] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[6] Requirements on Time Synchronization
AUTOSAR_FO_RS_TimeSync

[7] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[8] ISO/IEC 14882:2011, Information technology – Programming languages – C++
https://www.iso.org

[9] Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7
http://pubs.opengroup.org/onlinepubs/9699919799/

[10] Specification of Time Synchronization over Ethernet
AUTOSAR_CP_SWS_TimeSyncOverEthernet

[11] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

3.2 Further applicable specification

AUTOSAR provides a core specification [4] which is also applicable for this functional cluster. The chapter "General requirements for all Functional Clusters" of [4] shall be considered an additional and required specification for implementing this functional cluster.

4 Constraints and assumptions

4.1 Known limitations

The Time Synchronization module is bound to Adaptive Platform Systems.

4.1.1 Configuration

Please refer to the corresponding model elements.

4.1.2 Time Gateway

Time Gateway functionality is currently not in scope of the Time Synchronization module for the Adaptive Platform.

4.1.3 Out of Scope

Errors, which occurred during Global Time establishment and which are not caused by the module itself (i.e. loss of PTP global time is not an issue of the TS but of the TSP modules) are out of the scope of this module.

4.2 Applicability to car domains

The concept is targeted at supporting time-critical automotive applications. This does not mean that the concept has all that is required by such systems though, but crucial timing-related features which cannot be deferred to implementation are considered.

4.3 Recommendation

In the case where the TSP is based on Ethernet, the protocol to be used is defined in the PRS (see [1]).

...

5 Dependencies to other Functional Clusters

This chapter defines the dependencies of this functional cluster to other functional clusters. AUTOSAR decided not to standardize interfaces which are exclusively used between functional clusters to allow efficient implementations which might depend e.g., on the used operating system. The goal of this chapter is to provide an informative guideline for the interactions between functional clusters without specifying syntactical details. This ensures compatibility between documents specifying different functional clusters and supports parallel implementation of different functional clusters. Details of internal interfaces are up to the platform provider. Additional internal interfaces, parameters, and return values can be added. A detailed technical architecture documentation of the overall AUTOSAR Adaptive Platform is provided in [5].

5.1 Provided Interfaces

This section provides an overview of the public interfaces provided by this functional cluster towards other functional clusters.

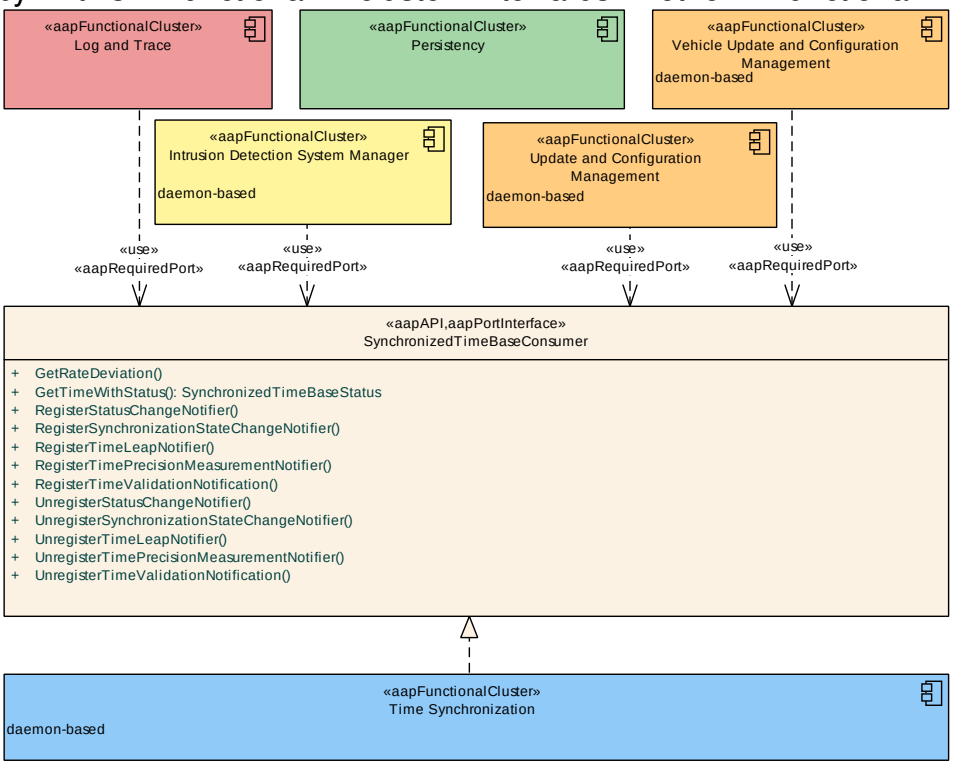

Figure 5.1: Interfaces provided by TimeSynchronization to other Functional Clusters

Figure 5.1 shows the interfaces provided by `Time Synchronization` to other functional clusters within the AUTOSAR Adaptive Platform. Table 5.1 lists the interfaces provided to other functional clusters within the AUTOSAR Adaptive Platform and provides a rationale.

Interface	Functional Cluster	Purpose
SynchronizedTimeBaseConsumer	Intrusion Detection System Manager	Adaptive Intrusion Detection System Manager shall use this interface to determine timestamps of security events.
	Log and Trace	Log and Trace shall use this interface to determine the timestamps that are associated with log messages.
	Raw Data Stream	This interface is used to determine the current synchronized global time for IEEE1722-based communication.
	Update and Configuration Management	Update and Configuration Management shall use this interface to get latest timestamp.
	Vehicle Update and Configuration Management	Vehicle Update and Configuration Management shall use this interface to get latest timestamp.

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

This section provides an overview of the public interfaces required by this functional cluster from other functional clusters.

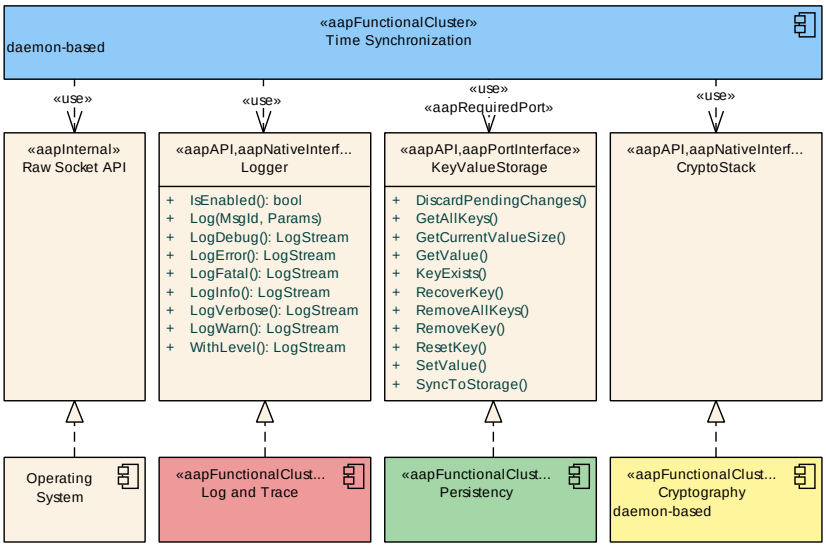

Figure 5.2: Interfaces required by Time Synchronization from other Functional Clusters

Figure 5.2 shows the interfaces required by Time Synchronization from other functional clusters within the AUTOSAR Adaptive Platform. Table 5.2 lists the interfaces required from other functional clusters within the AUTOSAR Adaptive Platform and provides a rationale.

Functional Cluster	Interface	Purpose
Cryptography	CryptoStack	Time Synchronization shall use this interface to generate / verify checksums (MAC).
Execution Management	ExecutionClient	Time Synchronization shall use this interface to report the state of its daemon process.

△

Functional Cluster	*Interface*	*Purpose*
Log and Trace	Logger	`Time Synchronization` shall use this interface to log standardized messages.
Persistency	KeyValueStorage	Used to store the last received timestamp to enable a faster startup.
Platform Health Management	SupervisedEntity	`Time Synchronization` should use this interface to enable supervision of its daemon process by `Platform Health Management`

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified in Time Synchronization [6] and the AUTOSAR RS General [7], and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_AP_00120]	Method and Function names	[SWS_TS_00903] [SWS_TS_00905] [SWS_TS_00906] [SWS_TS_00907] [SWS_TS_00908] [SWS_TS_00909] [SWS_TS_00910] [SWS_TS_01266] [SWS_TS_01267]
[RS_AP_00121]	Parameter names	[SWS_TS_00903] [SWS_TS_00907] [SWS_TS_00908] [SWS_TS_00910]
[RS_AP_00122]	Type names	[SWS_TS_00901] [SWS_TS_00902] [SWS_TS_00904]
[RS_AP_00127]	Usage of ara::core types	[SWS_TS_00901] [SWS_TS_00902] [SWS_TS_00904]
[RS_AP_00130]	AUTOSAR Adaptive Platform shall represent a rich and modern programming environment	[SWS_TS_00901] [SWS_TS_00902] [SWS_TS_00903] [SWS_TS_00904] [SWS_TS_00905] [SWS_TS_00906] [SWS_TS_00907] [SWS_TS_00908] [SWS_TS_00909] [SWS_TS_00910] [SWS_TS_01251] [SWS_TS_01260] [SWS_TS_01261] [SWS_TS_01262] [SWS_TS_01263] [SWS_TS_01264] [SWS_TS_01265] [SWS_TS_01266] [SWS_TS_01267]
[RS_AP_00150]	Provide only interfaces that are intended to be used by AUTOSAR Applications and Functional Clusters	[SWS_TS_00902] [SWS_TS_00904]
[RS_AP_00154]	Internal namespaces	[SWS_TS_00901] [SWS_TS_00902] [SWS_TS_00904] [SWS_TS_00909] [SWS_TS_00910]
[RS_Ids_00810]	Basic SW security events	[SWS_TS_14214] [SWS_TS_14215] [SWS_TS_14216] [SWS_TS_14217] [SWS_TS_14220]
[RS_TS_00002]	The Implementation of Time Synchronization shall maintain its own Time Base independently of the acting role.	[SWS_TS_00041] [SWS_TS_00042]
[RS_TS_00004]	The Implementation of Time Synchronization shall initialize the Global Time Base with a configurable startup value.	[SWS_TS_00213]
[RS_TS_00005]	The Implementation of Time Synchronization shall allow customers to have access to the Synchronized Time Base	[SWS_TS_01001] [SWS_TS_01002] [SWS_TS_01003] [SWS_TS_01004] [SWS_TS_01005] [SWS_TS_01006] [SWS_TS_01101] [SWS_TS_01102] [SWS_TS_01103] [SWS_TS_01104] [SWS_TS_01105] [SWS_TS_01106] [SWS_TS_01109] [SWS_TS_01251] [SWS_TS_01260] [SWS_TS_01261] [SWS_TS_01262] [SWS_TS_01263] [SWS_TS_01264] [SWS_TS_01265]

△

Requirement	Description	Satisfied by
[RS_TS_00007]	The Implementation of Time Synchronization shall synchronize the Time Base of a Time Slave, on reception of a Time Master value	[SWS_TS_00042] [SWS_TS_00055] [SWS_TS_00060]
[RS_TS_00009]	The Implementation of Time Synchronization shall maintain the synchronization status of a Time Base	[SWS_TS_00007] [SWS_TS_00011] [SWS_TS_00027] [SWS_TS_00028] [SWS_TS_00030] [SWS_TS_00032] [SWS_TS_00033] [SWS_TS_00055] [SWS_TS_00064] [SWS_TS_00139] [SWS_TS_00140] [SWS_TS_00141] [SWS_TS_00702] [SWS_TS_01050] [SWS_TS_01051] [SWS_TS_01108] [SWS_TS_14218]
[RS_TS_00010]	The Implementation of Time Synchronization shall allow customer on master side to set the Global Time	[SWS_TS_01107] [SWS_TS_01108]
[RS_TS_00011]	The Implementation of Time Synchronization shall allow customers on master side to trigger time transmission by the TSP module	[SWS_TS_01108]
[RS_TS_00014]	The Implementation of Time Synchronization shall allow customers to read User Data propagated via the TSP modules.	[SWS_TS_00120] [SWS_TS_01056] [SWS_TS_01113] [SWS_TS_14219]
[RS_TS_00015]	The Implementation of Time Synchronization shall allow customers to set User Data propagated via the TSP modules.	[SWS_TS_01112]
[RS_TS_00018]	The Implementation of Time Synchronization shall support rate correction	[SWS_TS_00041] [SWS_TS_00042] [SWS_TS_00043] [SWS_TS_00044] [SWS_TS_00045] [SWS_TS_00046] [SWS_TS_00048] [SWS_TS_00050] [SWS_TS_00051] [SWS_TS_00053] [SWS_TS_00054] [SWS_TS_00061] [SWS_TS_00062] [SWS_TS_00063] [SWS_TS_00070] [SWS_TS_00202] [SWS_TS_01008] [SWS_TS_01110] [SWS_TS_01111]
[RS_TS_00019]	The Implementation of Time Synchronization shall support damping offset correction	[SWS_TS_00042] [SWS_TS_00045] [SWS_TS_00050] [SWS_TS_00051] [SWS_TS_00054] [SWS_TS_00055] [SWS_TS_00056] [SWS_TS_00057] [SWS_TS_00058] [SWS_TS_00059]
[RS_TS_00021]	The Implementation of Time Synchronization shall provide interfaces to query the synchronization status	[SWS_TS_00120] [SWS_TS_00127] [SWS_TS_00701] [SWS_TS_01009] [SWS_TS_01052] [SWS_TS_01053] [SWS_TS_01054] [SWS_TS_01055] [SWS_TS_01056] [SWS_TS_01057] [SWS_TS_01058] [SWS_TS_01059] [SWS_TS_01060] [SWS_TS_01061] [SWS_TS_01062] [SWS_TS_01113] [SWS_TS_01403] [SWS_TS_14176] [SWS_TS_14219]
[RS_TS_00024]	The Implementation of Time Synchronization shall support storage of the Time Base value at shutdown if configured as Time Master	[SWS_TS_00212] [SWS_TS_00213] [SWS_TS_00214] [SWS_TS_00215]

▽

△

Requirement	Description	Satisfied by
[RS_TS_00026]	The Implementation of Time Synchronization shall provide to the customers a specific API per type of Time Base Resource	[SWS_TS_01107] [SWS_TS_01108] [SWS_TS_01109] [SWS_TS_01110] [SWS_TS_01112]
[RS_TS_00029]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a (vehicle wide) Time Master	[SWS_TS_00419] [SWS_TS_00421] [SWS_TS_00423] [SWS_TS_01108] [SWS_TS_01110] [SWS_TS_01112]
[RS_TS_00030]	The configuration of the Time Synchronization implementation shall allow the implementation to behave as a Time Slave	[SWS_TS_00420] [SWS_TS_00422] [SWS_TS_00428] [SWS_TS_01000] [SWS_TS_01016] [SWS_TS_01017] [SWS_TS_01100] [SWS_TS_01114] [SWS_TS_01115] [SWS_TS_01300] [SWS_TS_01301]
[RS_TS_00034]	The Implementation of Time Synchronization shall provide measurement data to the application	[SWS_TS_00414] [SWS_TS_00415] [SWS_TS_00416] [SWS_TS_00417] [SWS_TS_00419] [SWS_TS_00420] [SWS_TS_00421] [SWS_TS_00422] [SWS_TS_00423] [SWS_TS_00424] [SWS_TS_00425] [SWS_TS_00426] [SWS_TS_00427] [SWS_TS_00428] [SWS_TS_00703] [SWS_TS_00800] [SWS_TS_00803] [SWS_TS_01010] [SWS_TS_01011] [SWS_TS_01012] [SWS_TS_01013] [SWS_TS_01014] [SWS_TS_01015] [SWS_TS_01016] [SWS_TS_01017] [SWS_TS_01018] [SWS_TS_01019] [SWS_TS_01114] [SWS_TS_01115] [SWS_TS_01300] [SWS_TS_01301] [SWS_TS_01400] [SWS_TS_01401] [SWS_TS_01402] [SWS_TS_01403] [SWS_TS_01404] [SWS_TS_01405] [SWS_TS_01406] [SWS_TS_01407] [SWS_TS_01408] [SWS_TS_14140] [SWS_TS_14141] [SWS_TS_14142] [SWS_TS_14150] [SWS_TS_14151] [SWS_TS_14152] [SWS_TS_14153] [SWS_TS_14154] [SWS_TS_14155] [SWS_TS_14156] [SWS_TS_14160] [SWS_TS_14161] [SWS_TS_14162] [SWS_TS_14163] [SWS_TS_14164] [SWS_TS_14165] [SWS_TS_14166] [SWS_TS_14167] [SWS_TS_14170] [SWS_TS_14171] [SWS_TS_14172] [SWS_TS_14173] [SWS_TS_14174]
[RS_TS_20047]	The Timesync over Ethernet module shall trigger Time Base Synchronization transmission	[SWS_TS_14175]
[RS_TS_20058]	The Timesync over Ethernet module shall provide the precision of Synchronized Time Bases	[SWS_TS_14175]

▽

△

Requirement	Description	Satisfied by
[RS_TS_20072]	The Timesync over Ethernet module shall support means to secure the Time Synchronization protocol	[SWS_TS_14178] [SWS_TS_14179] [SWS_TS_14180] [SWS_TS_14181] [SWS_TS_14182] [SWS_TS_14183] [SWS_TS_14184] [SWS_TS_14185] [SWS_TS_14186] [SWS_TS_14187] [SWS_TS_14188] [SWS_TS_14189] [SWS_TS_14190] [SWS_TS_14191] [SWS_TS_14192] [SWS_TS_14193] [SWS_TS_14194] [SWS_TS_14195] [SWS_TS_14196] [SWS_TS_14197] [SWS_TS_14198] [SWS_TS_14199] [SWS_TS_14200] [SWS_TS_14201] [SWS_TS_14202] [SWS_TS_14203] [SWS_TS_14204] [SWS_TS_14205] [SWS_TS_14206] [SWS_TS_14207] [SWS_TS_14208] [SWS_TS_14209] [SWS_TS_14210] [SWS_TS_14211] [SWS_TS_14212] [SWS_TS_14213]

Table 6.1: Requirements Tracing

7 Functional specification

The functional behavior is described under the following specific contexts:

- Startup Behavior
- Shutdown Behavior
- Construction Behavior (Initialization)
- Normal Operation
- Error Handling
- Error Classification
- Version Check

7.1 General Overview of TS

For the Adaptive Platform, three different technologies were considered to fulfill such Time Synchronization requirements. These technologies were:

- StbM of the Classic Platform
- Library chrono - either std::chrono (C++11) or boost::chrono [8]
- The Time posix interface [9]

The following table shows the interfaces provided to the Application by means of this API and their equivalent interface in StbM.

Time Synchronization API - AP	***StbM - CP***
`GetTimeWithStatus`	StbM_GetCurrentTime
`SetTime`	StbM_SetGlobalTime
updateTime	StbM_UpdateGlobalTime
setUserData	StbM_SetUserData
getRateDeviation	StbM_GetRateDeviation
setRateCorrection	StbM_SetRateCorrection
timeLeap (attribute of the TimeBase Status class)	StbM_GetTimeLeap
getTimeBaseStatus	StbM_GetTimeBaseStatus
n/a	StbM_StartTimer
updateCounter (attribute of the TimeBase Status class)	StbM_GetTimeBaseUpdateCounter
This information is accessible via the Status flags	StbM_GetMasterConfig

Table 7.1: Interface comparison between TS and STBM

7.1.1 Base functionality of every Time Base

Every Time Base has to provide a minimum set of functionality, as listed below:

- offer possibility to obtain the current timestamp
- creating a snapshot of its parameters

This chapter briefly describes these functionalities. Details on how to use and the exact behavior of these core methods are given in chapter 8.

7.1.1.1 Time Base Status

This `TimeBaseStatus` is a snapshot of all the information of a Time Base Resource it is related to, like status flags, amount of times the TBR has been updated, time leap information (possibly generated during the last synchronization of the Time Base Resource), etc.

7.1.1.2 Rate Deviation

Applications will have different thresholds for acceptable time drift values. Hence there needs to be a way, how applications can access this information.

[SWS_TS_00202]

Upstream requirements: RS_TS_00018

⌈`ara::tsync::SynchronizedTimeBaseConsumer::GetRateDeviation` shall return the calculated rate deviation of its TBR against the time source it is synchronized to. In case there is no rate deviation calculated yet, the initial rate deviation of 0.0 shall be returned.⌋

Note: For more information of how rate deviation is calculated see: 7.2.6 Time Correction.

7.1.1.3 Clock Time Value

Reading the clock's time value is very likely the most commonly performed operation by the applications interacting with TS.

To ensure type safe handling of time values, the timepoint is provided as std::chrono structure.

More detailed information on how this is implemented is given in the further chapters and in chapter 8.

7.1.2 Status Flags of TBRs

Time Synchronization defines a set of status flags that are used to express specific status conditions of a TBR. Status flags can be queried by an application through a `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus`.

Synchronization status `GetSynchronizationStatus` includes:

- `kNotSynchronizedUntilStartup`: Indicates whether a synchronization of a time base to its corresponding TBR happend until start-up (initial state)
- `kTimeOut`: Indicates whether a synchronization of a time base to its corresponding TBR is lost or delayed.
- `kSynchronized`: Indicates if the time base of the corresponding TBR has been successfully synchronized at least once against its time source.
- `kSynchToGateway`: Indicates if the corresponding TBR updates are based on a Time Gateway below the Global Time Master.

The status if a leap jump happend since the last status request through a `GetTimeWithStatus` could be retrieved via `GetLeapJump`:

- `kTimeLeapNone`: Indicates that no leap jump happend
- `kTimeLeapFuture`: Indicates if there has been a jump in time to the future.
- `kTimeLeapPast`: Indicates if there has been a jump in time to the past.

7.1.3 Time Synchronization and Protocols

Time Synchronization mechanisms and protocols (i.e. [10] are out of the Scope of this document, for protocol specification please refer to the PRS (see [1]).

7.2 Normal Operation

7.2.1 Introduction

A Global Time network consists of a Time Master and at least one Time Slave. For each Time Domain, the Time Master is distributing the Global Time Base to the connected Time Slaves via Time Synchronization messages. The Time Slave corrects the received Global Time Base taking into account the Time Stamp at the transmitter side and the own generated receiver Time Stamp.

The local time of a Slave Time Base will be maintained autonomously and updated whenever a new time value is received from its associated Master Time Base.

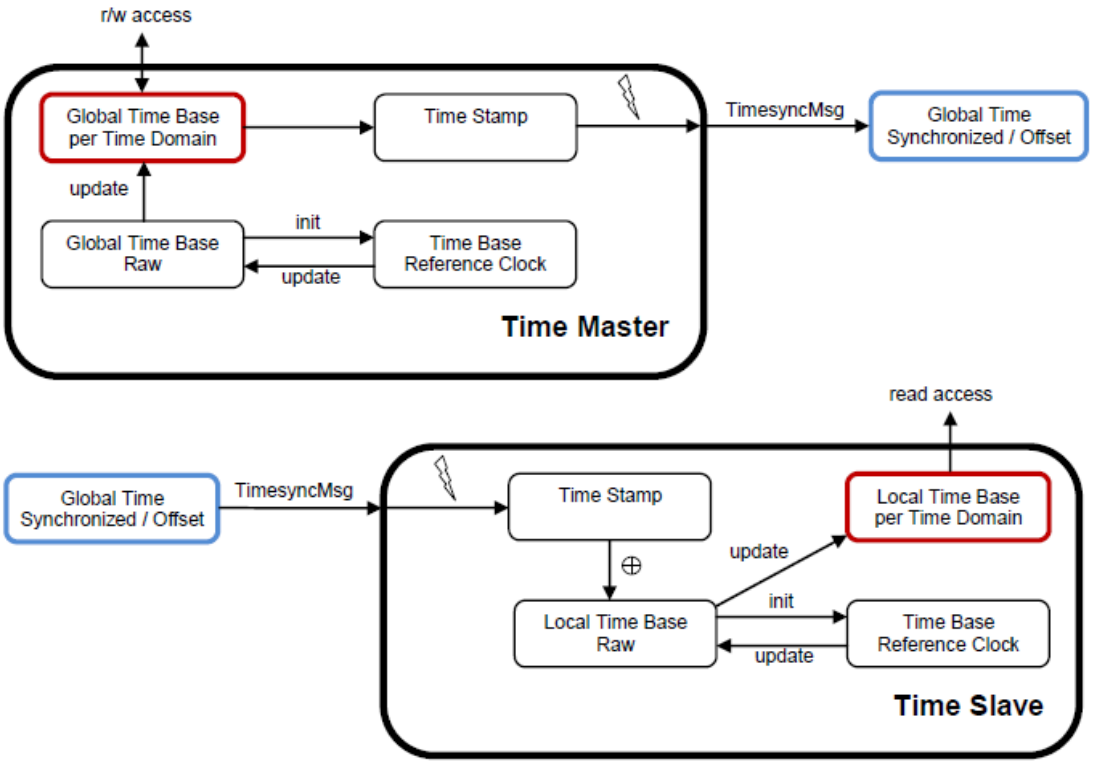

Figure 7.1: Global Time Base Distribution.

7.2.1.1 Time Base Manifestations

The number of Synchronized Time Bases is not limited by the TS functionality, but by the functional needs of the system to be fulfilled (i.e. the TS does not define a limit of Synchronized Time Bases identifiers in the system).

7.2.2 Roles of the Time Base Resources

7.2.2.1 Global Time Master

A TBR can act as a Global Time Master, in which case it is the system wide origin for a given time value that is then distributed via the network to the Time Slaves.

7.2.2.2 Time Slave

In the role of a Time Slave, the TBR updates its internally-maintained local time to a value of a Global Time Base, which is provided by the corresponding TSP module.

7.2.3 Time Base Resources

7.2.3.1 Slave Time Bases

[SWS_TS_00139]

Upstream requirements: RS_TS_00009

⌈Monitoring of time leaps to the future shall only be enabled, if a `timeLeapFutureThreshold` is other than zero and `ara::tsync::SynchronizationStatus` unequal to `kNotSynchronizedUntilStartup`.⌋

[SWS_TS_00140]

Upstream requirements: RS_TS_00009

⌈Monitoring of time leaps to the past shall only be enabled, if a `timeLeapPastThreshold` is other than zero and `ara::tsync::SynchronizationStatus` unequal to `kNotSynchronizedUntilStartup`.⌋

[SWS_TS_00141]

Upstream requirements: RS_TS_00009

⌈A check for time leaps shall be performed on every successful synchronization with the master clock, but only after the clock has been synchronized once (`ara::tsync::SynchronizationStatus` unequal to `kNotSynchronizedUntilStartup`).⌋

[SWS_TS_00027]

Upstream requirements: RS_TS_00009

⌈If the adjustment made by the resynchronization exceeded the specified threshold values, the corresponding `ara::tsync::LeapJump` status shall be set to `kTimeLeapNone` if no leap jump occurred. `kTimeLeapFuture`: if jump occured in time to the future greater than `timeLeapFutureThreshold`. `kTimeLeapPast`: if jump occured in time to the past greater than `timeLeapPastThreshold`.⌋

[SWS_TS_00064]

Upstream requirements: RS_TS_00009

⌈The initial value of `ara::tsync::LeapJump` shall be `kTimeLeapNone`.⌋

[SWS_TS_00028]

Upstream requirements: RS_TS_00009

⌈Active Time Leap Status `ara::tsync::LeapJump` shall be set to `kTimeLeapNone`, if a consecutive number `timeLeapHealingCounter` of synchronizations were all below the Time Leap Future and Past Thresholds.⌋

[SWS_TS_00030]

Upstream requirements: RS_TS_00009

⌈Each instance of `ara::tsync::SynchronizedTimeBaseConsumer` shall independently monitor for a synchronization timeout by measuring the time since that last update and a specified timeout duration in `syncLossTimeout`.⌋

[SWS_TS_00032]

Upstream requirements: RS_TS_00009

⌈In case of a monitored timeout (refer [SWS_TS_00030]) the `ara::tsync::SynchronizationStatus` shall be set to `kTimeOut`.⌋

[SWS_TS_00011]

Upstream requirements: RS_TS_00009

⌈If the update of the Time Base is successful and SYNC_TO_GATEWAY bit is set, the `ara::tsync::SynchronizationStatus` shall be set to `kSynchToGateway`.⌋

[SWS_TS_00033]

Upstream requirements: RS_TS_00009

⌈If the update of the Time Base is successful and the SYNC_TO_GATEWAY bit is NOT set, the `ara::tsync::SynchronizationStatus` shall be set to `kSynchronized`.⌋

[SWS_TS_14218] Get Time With Status

Upstream requirements: RS_TS_00009

⌈When `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus` is called for a Time Base, then `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus` shall sample the current Time Base time and status, i.e.,

- the current Global Time (synchronized time)
- the current Local Time (`ara::core::SteadyClock`)
- the current synchronization status
- the current time leap status
- the current user data

and return this information as a GetSynchronizationStatus object.⌋

[SWS_TS_00127]

Upstream requirements: RS_TS_00021

⌈For `ara::tsync::SynchronizedTimeBaseStatus` objects that correspond to a Synchronized TBR, `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus` shall return a copy of the same `ara::tsync::SynchronizedTimeBaseStatus` object this method belongs to.⌋

7.2.4 Immediate Time Synchronization

All TSP Modules are working independently of the TS regarding the handling of the bus-specific Time Synchronization protocol (i.e. autonomous transmission of Timesync messages on the bus).

Time information is passed from a TSP to the TBR. Implementation details as well as the interaction of such a TSP with the TBR are outside of the scope of this specification(for protocol specification please refer to [1]).

7.2.5 User Data

User Data is part of each Time Base. User Data are set by the system-wide Time Master of each Time Base and distributed as part of the Timesync messages.

User Data are used for application specific purposes e.g. :

- for debugging purposes during integration
- and/or to characterize the Time Base, e.g.,
  - regarding the quality of the underlying clock source
  - or regarding the progress of time (e.g., if Global Time has been restored from non-volatile memory after reset).

Note, that the status information, if contained in the user data, is from the Time Master's point of view, whereas the StatusFlag (`ara::tsync::SynchronizedTimeBaseStatus`) refers to the status of the Time Base from a Time Slave's point of view.

[SWS_TS_00120]

Upstream requirements: RS_TS_00014, RS_TS_00021

⌈In case there are no User Data stored, `ara::tsync::SynchronizedTimeBaseStatus::GetUserData` shall return an empty vector.⌋

[SWS_TS_14219] No user data available

Upstream requirements: RS_TS_00014, RS_TS_00021

⌈In case there are no User Data available, `ara::tsync::SynchronizedTimeBaseProvider` shall return an empty Vector.⌋

7.2.6 Time Correction

TS provides the ability for Time Slaves to perform Rate and Offset Correction of the Synchronized TBR.

For Global Time Masters, the TS provides the ability to perform Rate Correction of their Time Base(s).

Time correction can be configured individually for each Time Base.

7.2.6.1 Rate Correction for Time Slaves

Rate Correction detects and eliminates rate deviations of local instances of Time Bases. Rate Correction determines the rate deviation in the scope of a measurement. This rate deviation is used as correction factor which the TBR uses to correct the Time Base's time whenever it is read (e.g. in the scope of `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus`).

[SWS_TS_00041]

Upstream requirements: RS_TS_00002, RS_TS_00018

⌈The TBR shall perform Rate Correction measurements to determine its rate deviation if `ara::tsync::SynchronizationStatus` is set to `kSynchronized`.⌋

[SWS_TS_00042]

Upstream requirements: RS_TS_00002, RS_TS_00007, RS_TS_00018, RS_TS_00019

⌈The TBR shall perform Rate Correction measurements continuously. The end of a measurement marks the start of the next measurement.

The start and end of measurements is always triggered by (and aligned to) the reception of time values for Synchronized.⌋

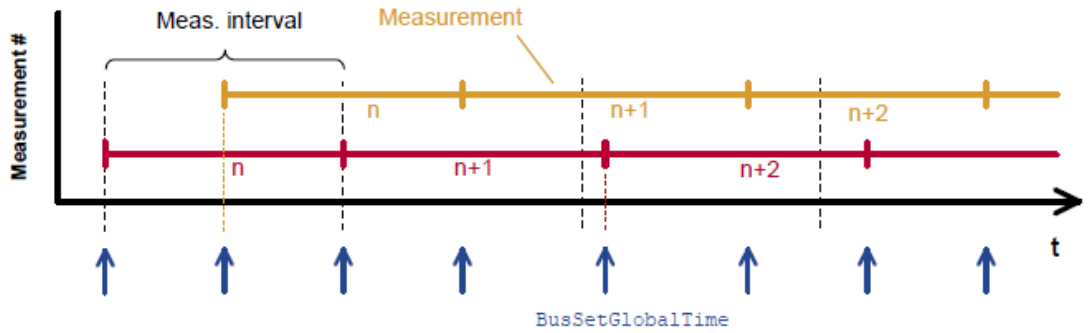

Figure 7.2: Visualization of two parallel measurements.

[SWS_TS_00043]

Upstream requirements: RS_TS_00018

⌈During runtime, the Synchronized TBR shall determine the timespan of a Rate Correction measurement on the basis of clock `ara::core::SteadyClock`.⌋

[SWS_TS_00044]

Upstream requirements: RS_TS_00018

⌈The TBR shall perform as many simultaneous Rate Correction measurements as configured by the parameter '`TimeSyncCorrection.rateCorrectionsPerMeasurementDuration`'.⌋

[SWS_TS_00045]

Upstream requirements: RS_TS_00018, RS_TS_00019

⌈Simultaneous Rate Correction measurements shall be started with a defined offset (to_n) to yield Rate Corrections evenly distributed over the measurement duration. The value will be calculated according to the following formula:
to_n = `n * (rateDeviationMeasurementDuration / rateCorrectionPerMeasurementDuration)` (where 'n' is the zero-based index of the current measurement)⌋

[SWS_TS_00046]

Upstream requirements: RS_TS_00018

⌈At the start of a Rate Correction measurement, the Synchronized TBR shall take the time-snapshots `TGStart` and `TVStart` in the scope of TSP.⌋

[SWS_TS_00048]

Upstream requirements: RS_TS_00018

⌈At the end of the Rate Correction measurement, the Synchronized TBR shall take the time-snapshots `TGStop` and `TVStop` in the scope TSP.⌋

[SWS_TS_00050]

Upstream requirements: RS_TS_00018, RS_TS_00019

⌈At the end of a Rate Correction measurement, the Synchronized TBR shall calculate the resulting correction rate (r_{rc}) according to the following formula:
$r_{rc} = (TG_{Stop} - TG_{Start}) / (TV_{Stop} - TV_{Start})$⌋

Note: To determine the resulting rate deviation the value 1 has to be subtracted from r_{rc}.

[SWS_TS_00051]

Upstream requirements: RS_TS_00018, RS_TS_00019

⌈The last r_{rc} value has to be used until a new value is calculated.⌋

[SWS_TS_00053]

Upstream requirements: RS_TS_00018

⌈On invocation of `ara::tsync::SynchronizedTimeBaseConsumer::GetRateDeviation` the TBR shall return the calculated rate deviation (i.e. r_{rc}-1).⌋

[SWS_TS_00070]

Upstream requirements: RS_TS_00018

⌈If no rate correction r_{rc} has yet been calculated, `ara::tsync::SynchronizedTimeBaseConsumer::GetRateDeviation` shall return 0.0.⌋

[SWS_TS_00054]

Upstream requirements: RS_TS_00018, RS_TS_00019

⌈If a valid correction rate (r_{rc}) has been calculated, the Synchronized TBR shall apply a Rate Correction.⌋

7.2.6.2 Offset Correction for Time Consumer

Offset Correction eliminates time offsets of local instances of Synchronized Time Bases. This correction takes place whenever the current time is read (e.g. in the scope of `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus`). The offset is measured when the local instance of the Time Base is synchronized in the scope of TSP.

[SWS_TS_00055]

Upstream requirements: RS_TS_00007, RS_TS_00009, RS_TS_00019

⌈For Synchronized TBRs, it shall be measured the offset between its local instance of the Time Base and the Global Time Base whenever the Time Base is synchronized in the scope of the function TSP by taking a snapshot of the TLSync and TVSync.⌋

[SWS_TS_00056]

Upstream requirements: RS_TS_00019

⌈If the absolute value of the time offset between Global Time Base and local instance of the Time Base (abs(TG - TL_{Sync})) is equal or greater than '`TimeSyncCorrection.offsetCorrectionJumpThreshold`', the TBR shall calculate the corrected time (TL) of its local instance of the Time Base according to the following formula:
$TL = TG + (TV - TV_{Sync}) * r_{rc}$⌋

Note:
This correction will be done whenever the time is read in the scope of e.g. the function `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus`.

Note:
This correction will be done when the TBR needs to determine the time of the local instance of the Time Base.

[SWS_TS_00057]

Upstream requirements: RS_TS_00019

⌈The TBR shall correct absolute time offsets between the Global Time Base and the local instance of the Time Base (abs(TG - TL_{Sync})), which are smaller than the value given by '`TimeSyncCorrection.offsetCorrectionJumpThreshold`' by temporarily applying an additional rate (r_{oc}) to r_{rc}. This rate shall be used for the duration defined by parameter '`TimeSyncCorrection.offsetCorrectionAdaptionInterval`'. r_{oc} is calculated according to the following formula:
$r_{oc} = (TG - TL_{Sync}) / (T_{CorrInt}) + 1$

⌋

[SWS_TS_00058]

Upstream requirements: RS_TS_00019

⌈If the absolute time offset between Global Time Base and local instance of the Time Base (abs(`TG` - TL_{Sync})) is smaller than '`TimeSyncCorrection.offsetCorrectionJumpThreshold`', the TBR shall calculate the corrected time (TL) of its local instance of the Time Base **within** the period of '`TimeSyncCorrection.offsetCorrectionAdaptionInterval`' according to the following formula:

$TL = TL_{Sync} + (r_{rc} * (TV - TV_{Sync}) * r_{oc})$

⌋

Note:
This correction will be done whenever the time is read in the scope of e.g. the function `ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus`.

Note:
This correction will be done when the TBR needs to determine the time of the local instance of the Time Base.

[SWS_TS_00059]

Upstream requirements: RS_TS_00019

⌈If the absolute time offset between the Global Time Base and the local instance of the Time Base (abs(TG - TL)) is smaller than `TimeSyncCorrection.offsetCorrectionJumpThreshold`, the TBR shall calculate the corrected time (TL) of its local instance of the Time Base **after** the period of `TimeSyncCorrection.offsetCorrectionAdaptionInterval` as specified in [SWS_TS_00056]⌋

[SWS_TS_00060]

Upstream requirements: RS_TS_00007

⌈If `TimeSyncCorrection.offsetCorrectionJumpThreshold` is set to 0, Offset Correction shall be performed by Jump Correction only.⌋

7.2.6.3 Rate Correction for Global Time Masters

Rate correction is applied by setting a correction factor which the TBR uses to correct the Time Base's time whenever it is transmitted over the network. This happens independent of the rate correction done by the slave.

[SWS_TS_00061]

Upstream requirements: RS_TS_00018

⌈

If `TimeSyncCorrection.allowProviderRateCorrection` equals *true*, and the rate deviation (= `rateCorrection` - 1.0) is within the valid range [-`TimeSyncCorrection.MasterRateDeviationMax` .. `TimeSyncCorrection.MasterRateDeviationMax`] , when `ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection` is called, TS shall set the rate correction value to the value of parameter `rateCorrection`.

If TimeSyncCorrection.allowProviderRateCorrection equals false, ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection TS shall not set a new rate correction value and shall return error code kFunctionNotSupported.⌋

Note: Refer [SWS_TS_00063] for handling of out of range rateCorrection values.

[SWS_TS_00062]

Upstream requirements: RS_TS_00018

⌈The TBR shall apply rate correction, if allowProviderRateCorrection equals TRUE and a valid rate correction value has been set by ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection.⌋

[SWS_TS_00063]

Upstream requirements: RS_TS_00018

⌈If allowProviderRateCorrection equals TRUE, and the rate correction greater than 1.0 + TimeSyncCorrection.providerRateDeviationMax, when ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection is called, TS shall set the rate correction value to 1.0 + TimeSyncCorrection.providerRateDeviationMax and ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection shall return kLimitsExceeded.

If allowProviderRateCorrection equals TRUE, and the rateCorrection is less than 1.0 -TimeSyncCorrection.providerRateDeviationMax, when ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection is called, TS shall set the rate correction value to 1.0 - TimeSyncCorrection.providerRateDeviationMax and SetRateCorrection shall return kLimitsExceeded.⌋

Note: When aligning the rate of one Time Base to the rate of another one, first read the rate deviation of the source Time Base via ara::tsync::SynchronizedTimeBaseProvider::GetRateDeviation and then pass the value (rate deviation + 1.0) as rateCorrection argument to ara::tsync::SynchronizedTimeBaseProvider::SetRateCorrection of the destination Time Base.

7.2.7 Notifications of Time Base Consumer

The Application might request to be notified of dedicated events for a specific TBR.

7.2.7.1 Status flags notification

A change in the StatusFlags of the `ara::tsync::SynchronizedTimeBaseStatus` can be notified.

[SWS_TS_00701]

Upstream requirements: RS_TS_00021

⌈A registered notifier via `ara::tsync::SynchronizedTimeBaseConsumer::RegisterStatusChangeNotifier` shall be invoked, if one of the following content is changed: `ara::tsync::SynchronizationStatus`, `ara::tsync::LeapJump` or the `user data`.⌋

7.2.7.2 Synchronization status notification

A change in the StatusFlags of the `ara::tsync::SynchronizationStatus` (e.g. if the timebase is in Timeout) can be notified.

[SWS_TS_00702]

Upstream requirements: RS_TS_00009

⌈A registered notifier via `ara::tsync::SynchronizedTimeBaseConsumer::RegisterSynchronizationStateChangeNotifier` shall be invoked, if `ara::tsync::SynchronizationStatus` is changed.⌋

7.2.7.3 LeapJump notification

A leap jump can be notified.

[SWS_TS_00703]

Upstream requirements: RS_TS_00034

⌈A registered notifier via `ara::tsync::SynchronizedTimeBaseConsumer::RegisterTimeLeapNotifier` shall be invoked, if `ara::tsync::LeapJump` is changed.⌋

7.2.8 Global Time Precision Measurement Support

To verify the precision of each Local Time Base compared to the Global Time Base a recording mechanism shall be optionally supported for Time Slaves and Time Gateways. In principle, a snapshot is taken of all required data at the point in time, where a synchronization event takes place. Access is provided to those values by an actively pushed API function on each successful assembled data block. An Off-Board Tester collects each block and calculates the precision afterwards and maintains a history of recorded blocks and their elements accordingly. How and by which protocol the data will be transferred to the Off-Board Tester will be specified by the Application.

[SWS_TS_00803]

Upstream requirements: RS_TS_00034

⌈A registration via `ara::tsync::SynchronizedTimeBaseConsumer::RegisterTimePrecisionMeasurementNotifier` shall only be possible for Synchronized Time Bases, for which `isSystemWideGlobalTimeMaster` is set to FALSE.⌋

[SWS_TS_00800]

Upstream requirements: RS_TS_00034

⌈For Synchronized Time Bases, a registered TimePrecisionMeasurement notifier (via `ara::tsync::SynchronizedTimeBaseConsumer::RegisterTimePrecisionMeasurementNotifier`) shall write the block elements

- `glbSeconds`
- `glbNanoSeconds`
- `timeBaseStatus`
- `virtualLocalTimeLow`
- `rateDeviation`
- `locSeconds`
- `locNanoSeconds`
- `pathDelay`

to the related measurement recording table after updating the Main Time Tuple (i.e., after updating the Local Time Base by the Global Time Base). GlbSeconds, GlbNanoSeconds are the elements of the Global Time part of the Received Time Tuple (i.e., TGRx); VirtualLocalTimeLow is the nanosecondsLo element of the Virtual Local Time part of the Received Time Tuple (i.e., TVRx).⌋

7.2.9 Global Time Validation Measurement Support

Figure 7.3 outlines the basic concept of the Time Validation feature.

A Time Slave collects information on the time synchronization process, to predict e.g. the Sync Ingress based on its local instance of Global Time and check whether Master and Slave agree upon the current time. The prediction itself will be locally analyzed by a separate Adaptive Application to detect any existing impairments. Furthermore, information on the time synchronization process from Time Masters and Slaves is also shared with a Validator Adaptive Application which may run anywhere in the network, e.g. on the owner of Global Time.

The Validator uses the information on the time synchronization process received from the Time Master and Time Slave Entities via a user defined feedback channel to reconstruct the whole synchronization process and check that a coherent time base is established among all peers.

The Time Validation feature only provides API to the Adaptive Application. The feedback channel and the actual validation performed by the respective Adaptive Application is not standardized in AUTOSAR. It is done in a user defined way on application level.

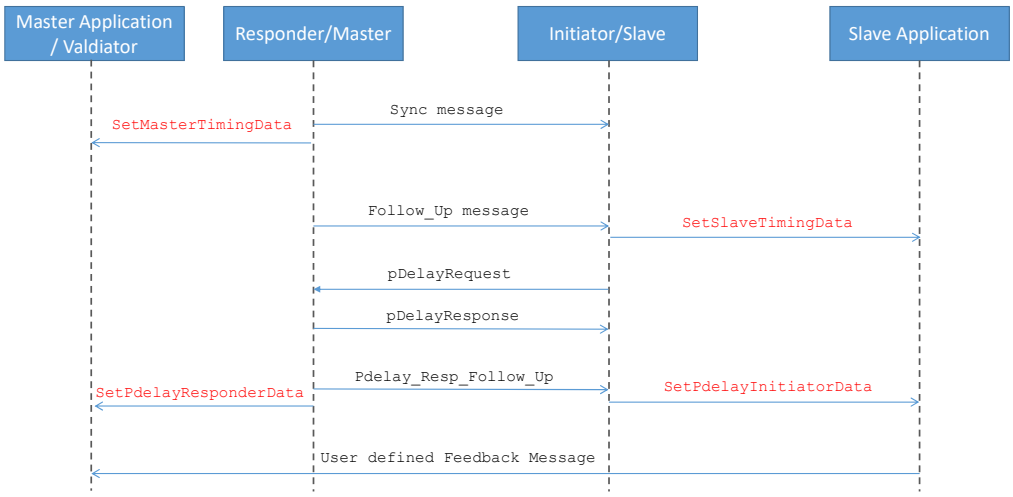

Figure 7.3: Time Validation mechanism

For an optional validation of the Timesync and Pdelay mechanisms, the Time Synchronization functional cluster provides the following functionality.

[SWS_TS_00424]

Upstream requirements: RS_TS_00034

⌈Everytime a `Follow_Up` message is received, all parameters defined by `ara::tsync::TimeSlaveMeasurementType` shall be updated and the function `ara::tsync::ConsumerTimeBaseValidationNotification::SetSlaveTimingData` shall be invoked.⌋

[SWS_TS_00425]

Upstream requirements: RS_TS_00034

⌈Everytime a Sync message is transmitted, all parameters defined by `ara::tsync::TimeMasterMeasurementType` shall be updated and the function `ara::tsync::ProviderTimeBaseValidationNotification::SetMasterTimingData` shall be invoked.⌋

[SWS_TS_00426]

Upstream requirements: RS_TS_00034

⌈After the current Pdelay measurement is finished, i.e., upon reception of the `Pdelay_Resp_Follow_Up` message, all parameters defined by `ara::tsync::PdelayInitiatorMeasurementType` shall be updated and the function `ara::tsync::ConsumerTimeBaseValidationNotification::SetPdelayInitiatorData` shall be invoked.⌋

[SWS_TS_00427]

Upstream requirements: RS_TS_00034

⌈After the current `Pdelay` measurement is finished, i.e., upon transmission of the `Pdelay_Resp_Follow_Up`, all parameters defined by `ara::tsync::PdelayResponderMeasurementType` shall be updated and the function `ara::tsync::ProviderTimeBaseValidationNotification::SetPdelayResponderData` shall be invoked.⌋

Note: Please note that there is a decoupling between reception and transmission of the respective PTP event messages and the forwarding of measurement data.

7.2.10 Secure Time Synchronization

7.2.10.1 Freshness value

The Freshness Value (FV) refers to a monotonic counter that is used to ensure freshness of the authenticated time synchronization messages. Freshness Values are to be derived from a Freshness Value Manager (FVM). FVM interface is defined by the appendix C.

[SWS_TS_14178] Initialization of the FVM

Upstream requirements: RS_TS_20072

⌈Freshness Value Manager shall be initialized prior to its usage by calling the `apext::tsync::FVM::Initialize`.⌋

7.2.10.2 Configuration of Secure TimeSync

[SWS_TS_14179] Usage of Secure Time Synchronization

Upstream requirements: RS_TS_20072

⌈Secure Time Synchronization shall be used if a SecOcSecureComProps instance is referenced in the role icvSecureComProps by a GlobalTimeDomain instance (refer to the description of GlobalTimeDomain in [11]).⌋

[SWS_TS_14180] Usage of Freshness Value

Upstream requirements: RS_TS_20072

⌈If the attribute GlobalTimeDomain.icvFreshnessValueId is set, the Freshness Value shall be used for the `ICV` calculation by Time Master. Time Slave shall check the attribute to verify if the received message is built properly, discard it in case its structure does not correspond to the configuration, and generate the security event SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE.⌋

[SWS_TS_14181] Check the configuration of Freshness Value

Upstream requirements: RS_TS_20072

⌈Time Slave shall check the status of the 'ICV with FV' bit of the AUTOSAR Sub-TLV:Time Authenticated of the TimeSync message (refer to PRS [1]). In case the Freshness Value shall be used, but the 'ICV with FV' bit is not set, Time Slave shall discard the message and generate the security event SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE.⌋

[SWS_TS_14182] Algorithm of ICV generation

Upstream requirements: RS_TS_20072

⌈The usage of the cryptographic algorithm for the `ICV` generation is defined by the icvSecureComProps.authentication element of the GlobalTimeDomain element (refer to the description of GlobalTimeDomain in [11]).⌋

7.2.10.3 Time Master

[SWS_TS_14183] Secure Time Synchronization message sending

Upstream requirements: RS_TS_20072

⌈The Secured Time Synchronization message shall be sent according to [1]. The following additional steps shall be performed:

- if the attribute icvFreshnessValueId is defined, obtain the `FV` by calling `apext::tsync::FVM::GetTxFreshness` and passing the configured freshnessValueId
- calculate the `ICV` using the obtained (optional) `FV`
- if the attribute icvFreshnessValueTxLength is defined, the `FV` shall be truncated and only the least significant bits shall be used in the message.
- construct the AUTOSAR TLV Sub-TLV: Time Authenticated with (optional) `FV` and `ICV` (see [1]).

⌋

[SWS_TS_14184] Secure Time Synchronization message build attempts

Upstream requirements: RS_TS_20072

⌈For every message to be sent and secured, an Authentication Build Counter shall be maintained:

- the Authentication Build Counter shall be set to 0 if the operation was successful
- if the query of the FV `apext::tsync::FVM::GetTxFreshness` returns a recoverable error kFVNotAvailable, or an error occurs during calculation of the ICV, the Authentication Build Counter is incremented and the ICV generation will be retried
- if the Authentication Build Counter has reached the value of the parameter authenticationBuildAttempts, the Time Master shall stop the ICV generation and accordingly set the ICV_flags in AUTOSAR TLV Sub-TLV: Time Authenticated of Follow_Up message.

⌋

7.2.10.4 Time Slave

[SWS_TS_14185] Secure Time Synchronization message receiving

Upstream requirements: RS_TS_20072

⌈The Secured Time Synchronization message shall be verified according to [1]. The following steps shall be performed:

- if the 'ICV with FV' bit is set in ICV_flags of the received Follow_Up message, and the attribute freshnessValueTxLength is defined, the FV shall be calculated by calling apext::tsync::FVM::GetRxFreshness with FreshnessValueId and the extracted truncated FV from the message, otherwise the FV shall be extracted from the message itself.
- the full ICV shall be extracted from the message
- verify the message by calculating the ICV using the extracted (optional) FV, and comparing the result with received ICV

⌋

[SWS_TS_14186] Secure Time Synchronization verification attempts

Upstream requirements: RS_TS_20072

⌈For every message received and secured, an ICV verification attempt counter shall be maintained:

- the ICV verification attempt counter shall be set to 0 if the operation was successful
- if the query of the FV by apext::tsync::FVM::GetRxFreshness call returns a recoverable error kFVNotAvailable, or an error occurs during calculation of the ICV, the ICV verification attempt counter shall be incremented and the process of message verification will be retried
- if the counter has reached the value of the parameter authenticationVerifyAttempts, the verification shall be stopped, the received message shall be discarded
- if the calculation of the ICV was successful but the verification failed, the message shall be discarded

⌋

7.3 Functional cluster life cycle

This section defines behavior of this functional cluster during its life-cycle. Please note that there is a general behavior for ara::core::Initialize and ara::core::Deinitialize defined in [4] by [SWS_CORE_90021] and [SWS_CORE_90022].

7.3.1 Startup

This chapter describes the necessary initializations, which are performed by the entity that has control over the Time Base Resources, in order to prepare the TS module for

normal operation. After its initialization, the module is expected to provide all synchronized time services to the applications.

[SWS_TS_14175]

Upstream requirements: RS_TS_20047, RS_TS_20058

⌈A TBR configured as Time Master shall start the transmission of sync message only if the network is available (link is up) and in one of the following conditions:

- a valid time is set or updated by the application or
- the backupTimeStamp is successfully restored from persistent memory

⌋

[SWS_TS_00213]

Upstream requirements: RS_TS_00024, RS_TS_00004

⌈For each TBR configured as Time Master for which storage to persistent memory is activated, i.e. a 'TimeBaseProviderToPersistencyMapping.timeBaseProvider' is present, the value of Global Time shall be restored from persistent memory such that the value of backupTimestamp is used to initialize the Time Base. Immediately after successfully loading the stored backupTimestamp, the Time Master shall store a new backupTimestamp ($= loaded(old) backupTimestamp +$ TimeBaseProviderToPersistencyMapping.cyclicBackupInterval)⌋

[SWS_TS_00214]

Upstream requirements: RS_TS_00024

⌈In case the restore from persistent memory is not successful, the Time Base shall start with zero.⌋

[SWS_TS_00215]

Upstream requirements: RS_TS_00024

⌈For each TBR configured as Time Slave, Clock Update Counter shall be initialized with zero.⌋

7.3.1.1 Default values

When the system starts up, the TBRs have to be set to known default values so that their behavior is well defined.

[SWS_TS_00007]

Upstream requirements: RS_TS_00009

⌈Characteristics of Time Base Resources shall be initialized as follows:

- Active Status Flags shall be invalidated.
- The User Data is to be deleted.
- Time Leap information shall be reset.

⌋

7.3.2 Shutdown

[SWS_TS_00212]

Upstream requirements: RS_TS_00024

⌈For each TBR configured as Time Master for which storage to persistent memory is activated, i.e a 'TimeBaseProviderToPersistencyMapping.timeBaseProvider' is present, first the current value of the Global Time shall be read every 'TimeBaseProviderToPersistencyMapping.cyclicBackupInterval'. Then the value of the 'TimeBaseProviderToPersistencyMapping.cyclicBackupInterval' itself shall be added and this check-pointed value of the Global Time shall then be stored into persistent memory 'TimeBaseProviderToPersistencyMapping.timeBaseProvider' (see [11]) as backupTimestamp if persistent storage is required. The initial value of backupTimestamp shall be set to 0.

Upon a graceful shutdown the Global Time shall be stored without applying another 'TimeBaseProviderToPersistencyMapping.cyclicBackupInterval' as back-off.⌋

Note: Regardless of the exact shutdown event, the last stored value of backupTimestamp will be restored during the next startup (see 7.3.1).

7.3.3 Daemon crash

A crash of the daemon will either be handled by the error code ara::tsync::TsyncErrc.kDaemonConnectionLost (mainly for Setter-APIs) or similar to message loss (resulting in ara::tsync::SynchronizationStatus.kTimeOut).

7.4 Reporting

7.4.1 Security Events

This chapter contains all standardized Security Events of this Functional Cluster.

[SWS_TS_14220] Security events for Time Synchronization (TS)

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

Name	*Description*	*ID*
SEV_TSYN_ETH_ICV_GENERATION_FAILED	ICV generation for a Follow_Up message failed.	73
SEV_TSYN_ETH_ICV_VERIFICATION_FAILED	ICV verification of a received Follow_Up message failed.	74
SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE	Failed to get freshness value from FvM.	75
SEV_TSYN_ETH_MSG_SEQUENCE_ERROR	Failed to receive correct sequence of SYNC and FUP from the TimeMaster within (EthTSynGlobalTimeFollowUp Timeout).	76

⌋

[SWS_TS_14187] Authentication Build Counter reached the threshold

Upstream requirements: RS_TS_20072

⌈If the Time Master is unable to generate the ICV and the Authentication Build Counter has reached the implementation specific threshold, the Time Master shall report the security event SEV_TSYN_ETH_ICV_GENERATION_FAILED to IdsM.⌋

[SWS_TS_14214] Security event context data definition: SEV_TSYN_ETH_ICV_GENERATION_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_TSYN_ETH_ICV_GENERATION_FAILED*	
ID	73	
Description	ICV generation for a Follow_Up message failed.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
GlobalTimeDomainId	uint8	

⌋

[SWS_TS_14188] Authentication Verify Counter reached the threshold

Upstream requirements: RS_TS_20072

⌈If the Time Slave is unable to verify the ICV and the ICV verification attempt counter has reached the value of the parameter authenticationVerifyAttempts, the Time Slave shall report the security event SEV_TSYN_ETH_ICV_VERIFICATION_FAILED to IdsM.⌋

[SWS_TS_14215] Security event context data definition: SEV_TSYN_ETH_ICV_VERIFICATION_FAILED

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_TSYN_ETH_ICV_VERIFICATION_FAILED***	
ID	74	
Description	ICV verification of a received Follow_Up message failed.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GlobalTimeDomainId	uint8	

⌋

[SWS_TS_14189] Freshness Value Manager is not available

Upstream requirements: RS_TS_20072

⌈If the query of the FV `apext::tsync::FVM::GetTxFreshness` returns a recoverable error kFVNotAvailable, the Time Master shall report the security event SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE to IdsM.⌋

[SWS_TS_14216] Security event context data definition: SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE***	
ID	75	
Description	Failed to get freshness value from FvM.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GlobalTimeDomainId	uint8	

⌋

[SWS_TS_14190] RxDebounceCounter value is greater than 0

Upstream requirements: RS_TS_20072

⌈If RxDebounceCounter is greater than 0 upon receiving the next TimeSync message, the Time Slave shall report the security event SEV_TSYN_ETH_MSG_SEQUENCE_ERROR to IdsM.⌋

[SWS_TS_14217] Security event context data definition: SEV_TSYN_ETH_MSG_SEQUENCE_ERROR

Status: DRAFT

Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_TSYN_ETH_MSG_SEQUENCE_ERROR***	
ID	76	
Description	Failed to receive correct sequence of SYNC and FUP from the TimeMaster within (EthTSyn GlobalTimeFollowUpTimeout).	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
GlobalTimeDomainId	uint8	

⌋

7.4.2 Log Messages

This chapter contains all Log Messages (i.e. DLT messages) of this Functional Cluster.

[SWS_TS_14191] LogMessage Authentication Build Counter has reached the threshold value

Upstream requirements: RS_TS_20072

⌈Whenever the Authentication Build Counter has reached the value of the parameter authenticationBuildAttempts (see [SWS_TS_14184]), Time Synchronization shall log a DltMessage of type FailedSecureTSynBuildAttempt with arguments set to:

- `AuthenticationBuildCounter_value_reached`: the value of the Authentication Build Counter.

⌋

[SWS_TS_14192] LogMessage ICV verification attempt counter has reached the threshold value

Upstream requirements: RS_TS_20072

⌈Whenever the ICV verification attempt counter has reached the value of the parameter authenticationVerifyAttempts (see [SWS_TS_14186]), Time Synchronization shall log a DltMessage of type FailedSecureTSynVerifyAttempt with arguments set to:

- `AuthenticationVerifyCounter_value_reached`: the value of the ICV verification attempt counter.

⌋

[SWS_TS_14193] LogMessage Verification of ICV failed

Upstream requirements: RS_TS_20072

⌈Whenever the verification of ICV failed (see [SWS_TS_14186]), Time Synchronization shall log a DltMessage of type FailedSecureTSynVerification.⌋

8 API specification

This chapter provides a reference of the APIs defined by this functional cluster. The API is described in the following chapters in tables. Table 8.1 explains the content that is described in such an API table.

Kind:	Defines the kind of the declaration that this API table describes. The following values are supported: • class (Declaration of a class) • function (Declaration of a member or non-member function) • struct (Declaration of a structure) • type alias (Declaration of a type alias) • enumeration (Declaration of an enumeration) • variable (Declaration of a variable)	
Header File:	Defines the header file to be included according to [SWS_CORE_90001]	
Forwarding Header File:	Defines the forwarding header file to be included according to [SWS_CORE_90001]	
Scope:	Defines the scope that may be a namespace (in case of a class or non-member function) or a class declaration (in case of a member)	
Symbol:	Entity name	
Thread Safety:	Defines whether a function is thread-safe, not thread-safe, or conditional according to [SWS_CORE_13200] and [SWS_CORE_13202]	
Syntax:	Description of C++ syntax	
Template Param:	Template parameter (0..*)	Template parameter(s) used to parametrize the template
Parameters (in):	Parameter declaration (0..*)	Parameter(s) that are passed to the function
Parameters (out):	Parameter declaration (0..*)	Parameter(s) that are returned to the caller
Return Value:	Return type	Type of the value that the function returns
Exception Safety:	Defines whether a function is exception-safe, not exception safe or conditionally exception safe	
Exceptions:	List of exceptions that may be thrown from the function	
Violations:	List of violations that may occur in the function	
Errors:	Error type (0..*)	List of defined error codes that may be returned by the function with their recoverability class defined in [RS_AP_00160]. APIs can be extended with vendor-specific error codes. These are not part of the AUTOSAR SWS specifications
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

8.1 Header: ara/tsync/consumer_time_base_validation_notification.h

8.1.1 Class: ConsumerTimeBaseValidationNotification

[SWS_TS_00428] Definition of API class ara::tsync::ConsumerTimeBaseValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	class
Header file:	#include "ara/tsync/consumer_time_base_validation_notification.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	ConsumerTimeBaseValidationNotification
Syntax:	`class ConsumerTimeBaseValidationNotification {...};`
Description:	Callback interface to notify Consumer Application about the availability of a new data block recorded for the Time Base. .

⌋

8.1.1.1 Public Member Functions

8.1.1.1.1 Special Member Functions

8.1.1.1.1.1 Destructor

[SWS_TS_01300] Definition of API function ara::tsync::ConsumerTimeBaseValidationNotification::~ConsumerTimeBaseValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function
Header file:	#include "ara/tsync/consumer_time_base_validation_notification.h"
Scope:	`class ara::tsync::ConsumerTimeBaseValidationNotification`
Syntax:	`virtual ~ConsumerTimeBaseValidationNotification () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Destructor .

⌋

8.1.1.1.2 Member Functions

8.1.1.1.2.1 SetPdelayInitiatorData

[SWS_TS_00422] Definition of API function ara::tsync::ConsumerTimeBaseValidationNotification::SetPdelayInitiatorData

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function	
Header file:	#include "ara/tsync/consumer_time_base_validation_notification.h"	
Scope:	`class ara::tsync::ConsumerTimeBaseValidationNotification`	
Syntax:	`virtual void SetPdelayInitiatorData (const PdelayInitiatorMeasurementType &measurementData) noexcept=0;`	
Parameters (in):	measurementData	Detailed timing data for the pDelay Initiator
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Provide the recorded data block for the pPelay Initiator of the Time Base.	

⌋

8.1.1.1.2.2 SetSlaveTimingData

[SWS_TS_00420] Definition of API function ara::tsync::ConsumerTimeBaseValidationNotification::SetSlaveTimingData

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function	
Header file:	#include "ara/tsync/consumer_time_base_validation_notification.h"	
Scope:	`class ara::tsync::ConsumerTimeBaseValidationNotification`	
Syntax:	`virtual void SetSlaveTimingData (const TimeSlaveMeasurementType &measurementData) noexcept=0;`	
Parameters (in):	measurementData	Detailed data for validation of the Time Slave
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Provide the recorded data block for the Time Slave of the Time Base.	

⌋

8.2 Header: ara/tsync/provider_time_base_validation_notification.h

8.2.1 Class: ProviderTimeBaseValidationNotification

[SWS_TS_00419] Definition of API class ara::tsync::ProviderTimeBaseValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00029

⌈

Kind:	class
Header file:	#include "ara/tsync/provider_time_base_validation_notification.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	ProviderTimeBaseValidationNotification
Syntax:	`class ProviderTimeBaseValidationNotification {...};`
Description:	Callback interface to notify (Validator) Application about the availability of a new data block recorded for the Time Base. gfddfgfdg

⌋

8.2.1.1 Public Member Functions

8.2.1.1.1 Special Member Functions

8.2.1.1.1.1 Destructor

[SWS_TS_01301] Definition of API function ara::tsync::ProviderTimeBaseValidationNotification::~ProviderTimeBaseValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function
Header file:	#include "ara/tsync/provider_time_base_validation_notification.h"
Scope:	`class ara::tsync::ProviderTimeBaseValidationNotification`
Syntax:	`virtual ~ProviderTimeBaseValidationNotification () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Destructor .

⌋

8.2.1.1.2 Member Functions

8.2.1.1.2.1 SetMasterTimingData

[SWS_TS_00421] Definition of API function ara::tsync::ProviderTimeBaseValidationNotification::SetMasterTimingData

Upstream requirements: RS_TS_00034, RS_TS_00029

⌈

Kind:	function	
Header file:	#include "ara/tsync/provider_time_base_validation_notification.h"	
Scope:	`class ara::tsync::ProviderTimeBaseValidationNotification`	
Syntax:	`virtual void SetMasterTimingData (const TimeMasterMeasurementType &measurementData) noexcept=0;`	
Parameters (in):	measurementData	Detailed data for validation of the Time Master
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Provide the recorded data block for the Time Master of the Time Base.	

⌋

8.2.1.1.2.2 SetPdelayResponderData

[SWS_TS_00423] Definition of API function ara::tsync::ProviderTimeBaseValidationNotification::SetPdelayResponderData

Upstream requirements: RS_TS_00034, RS_TS_00029

⌈

Kind:	function	
Header file:	#include "ara/tsync/provider_time_base_validation_notification.h"	
Scope:	`class ara::tsync::ProviderTimeBaseValidationNotification`	
Syntax:	`virtual void SetPdelayResponderData (const PdelayResponderMeasurementType &measurementData) noexcept=0;`	
Parameters (in):	measurementData	Detailed timing data for the pDelay Responder
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Provide the recorded data block for the pPelay Responder of the Time Base.	

⌋

8.3 Header: ara/tsync/synchronized_time_base_consumer.h

8.3.1 Class: SynchronizedTimeBaseConsumer

[SWS_TS_01000] Definition of API class ara::tsync::SynchronizedTimeBaseConsumer

Upstream requirements: RS_TS_00030

⌈

Kind:	class
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	SynchronizedTimeBaseConsumer
Syntax:	`class SynchronizedTimeBaseConsumer final {...};`
Description:	Class SynchronizedTimeBaseConsumer is the access to the synchronized timebase referenced by the IntstanceSpecifier. It allows to get the current time_point, the rate deviation, the current status and the received user data

⌋

8.3.1.1 Public Member Functions

8.3.1.1.1 Special Member Functions

8.3.1.1.1.1 Copy Constructor

[SWS_TS_01005] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::SynchronizedTimeBaseConsumer

Upstream requirements: RS_TS_00005

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`SynchronizedTimeBaseConsumer (const SynchronizedTimeBaseConsumer &)=delete;`
Description:	The copy constructor for SynchronizedTimeBaseConsumer shall not be used.

⌋

8.3.1.1.1.2 Move Constructor

[SWS_TS_01003] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer

Upstream requirements: RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`SynchronizedTimeBaseConsumer (SynchronizedTimeBaseConsumer &&stbc) noexcept;`	
Parameters (in):	stbc	The SynchronizedTimeBaseConsumer object to be moved.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor for SynchronizedTimeBaseConsumer.	

⌋

8.3.1.1.1.3 Copy Assignment Operator

[SWS_TS_01006] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::operator=

Upstream requirements: RS_TS_00005

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`SynchronizedTimeBaseConsumer & operator= (const SynchronizedTimeBase Consumer &)=delete;`
Description:	The copy assignment operator for SynchronizedTimeBaseConsumer shall not be used.

⌋

8.3.1.1.1.4 Move Assignment Operator

[SWS_TS_01004] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::operator=

Upstream requirements: RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`SynchronizedTimeBaseConsumer & operator= (SynchronizedTimeBaseConsumer &&stbc) & noexcept;`	
Parameters (in):	stbc	The SynchronizedTimeBaseConsumer object to be moved.
Return value:	SynchronizedTimeBase Consumer &	The moved SynchronizedTimeBaseConsumer object.
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Move assignment operator for SynchronizedTimeBaseConsumer. *	

⌋

8.3.1.1.1.5 Destructor

[SWS_TS_01002] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::~SynchronizedTimeBaseConsumer

Upstream requirements: RS_TS_00005

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`~SynchronizedTimeBaseConsumer () noexcept;`
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	SynchronizedTimeBaseConsumer destructor.

⌋

8.3.1.1.2 Constructors

8.3.1.1.2.1 SynchronizedTimeBaseConsumer

[SWS_TS_01001] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::SynchronizedTimeBaseConsumer

Upstream requirements: RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`explicit SynchronizedTimeBaseConsumer (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to a PortPrototype typed by a SynchronizedTimeBaseConsumerInterface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `TimeSyncPortPrototypeToTimeBaseMapping` needs to be typed by a `SynchronizedTimeBaseConsumerInterface`.
Description:	SynchronizedTimeBaseConsumer constructor.	

⌋

8.3.1.1.3 Member Functions

8.3.1.1.3.1 GetRateDeviation

[SWS_TS_01008] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::GetRateDeviation

Upstream requirements: RS_TS_00018

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`double GetRateDeviation () const noexcept;`	
Return value:	double	The current rate deviation.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Method to obtain the current rate deviation of the clock.	

⌋

8.3.1.1.3.2 GetTimeWithStatus

[SWS_TS_01009] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::GetTimeWithStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`SynchronizedTimeBaseStatus GetTimeWithStatus () const noexcept;`	
Return value:	SynchronizedTimeBaseStatus	A clock specific TimeBaseStatus that contains all the relevant clock information.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Method to obtain a snapshot of the current state of the clock. This includes status flags, clock configuration and the actual time value (local and synchronized) of the created status object.	

⌋

8.3.1.1.3.3 RegisterStatusChangeNotifier

[SWS_TS_01010] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::RegisterStatusChangeNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`void RegisterStatusChangeNotifier (std::function< void(const SynchronizedTimeBaseStatus &)> notifier) noexcept;`	
Parameters (in):	notifier	The notifier function to be registered.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Register a notifier function which is called if a StatusFlag is changed (i.e. synchronization state, time leap or userdata). A maximum of one notifier can be registered. Every further registration overwrites the current registration.	

⌋

8.3.1.1.3.4 RegisterSynchronizationStateChangeNotifier

[SWS_TS_01012] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::RegisterSynchronizationStateChangeNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`void RegisterSynchronizationStateChangeNotifier (std::function< void(const SynchronizationStatus &)> notifier) noexcept;`	
Parameters (in):	notifier	The function to unregister.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Register a notifier function which is called if a synchronization state is changed. A maximum of one notifier can be registered. Every further registration overwrites the current registration.	

⌋

8.3.1.1.3.5 RegisterTimeLeapNotifier

[SWS_TS_01014] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::RegisterTimeLeapNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`void RegisterTimeLeapNotifier (std::function< void(const SynchronizedTimeBaseStatus &)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called if the TimeBaseStatus has changed.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Register a notifier function which is called if a time leap happend. A maximum of one notifier can be registered. Every further registration overwrites the current registration.	

⌋

8.3.1.1.3.6 RegisterTimePrecisionMeasurementNotifier

[SWS_TS_01018] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::RegisterTimePrecisionMeasurementNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`void RegisterTimePrecisionMeasurementNotifier (std::function< void(const TimePrecisionMeasurement &)> notifier) noexcept;`	
Parameters (in):	notifier	The function to be called.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Register a notifier function which is called if a new time precision snapshot is available. A maximum of one notifier can be registered. Every further registration overwrites the current registration. The Tsync will not do any queuing. If needed it has to be done within the notifier.	

⌋

8.3.1.1.3.7 RegisterTimeValidationNotification

[SWS_TS_01016] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::RegisterTimeValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`	
Syntax:	`void RegisterTimeValidationNotification (ConsumerTimeBaseValidationNotification &timeBaseValidationNotification) noexcept;`	
Parameters (in):	timeBaseValidationNotification	time consumer application notification object that will be notified about the availability of a new data block recorded for the Time Base
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Method that can be used by time consumer applications to receive time sync parameters. A maximum of one notifier can be registered. Every further registration overwrites the current registration.	

⌋

8.3.1.1.3.8 UnregisterStatusChangeNotifier

[SWS_TS_01011] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::UnregisterStatusChangeNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`void UnregisterStatusChangeNotifier () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Unregister a notifier function which is called if a StatusFlag is changed (i.e. synchronization state, time leap or userdata). .

⌋

8.3.1.1.3.9 UnregisterSynchronizationStateChangeNotifier

[SWS_TS_01013] Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::UnregisterSynchronizationStateChangeNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`void UnregisterSynchronizationStateChangeNotifier () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Unregister a notifier function which is called if a synchronization state is changed. .

⌋

8.3.1.1.3.10 UnregisterTimeLeapNotifier

[SWS_TS_01015] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterTimeLeapNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`void UnregisterTimeLeapNotifier () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Unregister a notifier function which is called if a time leap happend. .

⌋

8.3.1.1.3.11 UnregisterTimePrecisionMeasurementNotifier

[SWS_TS_01019] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterTimePrecisionMeasurementNotifier

Upstream requirements: RS_TS_00034

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`void UnregisterTimePrecisionMeasurementNotifier () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Unregister a notifier function which is called if a new time precision snapshot is available. .

⌋

8.3.1.1.3.12 UnregisterTimeValidationNotification

[SWS_TS_01017] Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterTimeValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_consumer.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseConsumer`
Syntax:	`void UnregisterTimeValidationNotification () noexcept;`
Return value:	None
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Method that can be used by time consumer applications to receive time sync parameters. .

⌋

8.4 Header: ara/tsync/synchronized_time_base_provider.h

8.4.1 Class: SynchronizedTimeBaseProvider

[SWS_TS_01100] Definition of API class ara::tsync::SynchronizedTimeBase Provider

Upstream requirements: RS_TS_00030

⌈

Kind:	class
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	SynchronizedTimeBaseProvider
Syntax:	`class SynchronizedTimeBaseProvider final {...};`
Description:	Class SynchronizedTimeBaseProvider is the access to the synchronized timebase referenced by the IntstanceSpecifier. It allows to get the current time_point, the rate deviation, the current status and the received user data

⌋

8.4.1.1 Public Member Functions

8.4.1.1.1 Special Member Functions

8.4.1.1.1.1 Move Constructor

[SWS_TS_01102] Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider

Upstream requirements: RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`SynchronizedTimeBaseProvider (SynchronizedTimeBaseProvider &&stbc) noexcept;`	
Parameters (in):	stbc	The SynchronizedTimeBaseProvider object to be moved.
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Move constructor for SynchronizedTimeBaseProvider.	

⌋

8.4.1.1.1.2 Copy Constructor

[SWS_TS_01104] Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider

Upstream requirements: RS_TS_00005

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`
Syntax:	`SynchronizedTimeBaseProvider (const SynchronizedTimeBaseProvider &)=delete;`
Description:	The copy constructor for SynchronizedTimeBaseProvider shall not be used.

⌋

8.4.1.1.1.3 Copy Assignment Operator

[SWS_TS_01105] Definition of API function ara::tsync::SynchronizedTimeBase Provider::operator=

Upstream requirements: RS_TS_00005

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`
Syntax:	`SynchronizedTimeBaseProvider & operator= (const SynchronizedTimeBase Provider &)=delete;`
Description:	The copy assignment operator for SynchronizedTimeBaseProvider shall not be used.

⌋

8.4.1.1.1.4 Move Assignment Operator

[SWS_TS_01103] Definition of API function ara::tsync::SynchronizedTimeBase Provider::operator=

Upstream requirements: RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`SynchronizedTimeBaseProvider & operator= (SynchronizedTimeBaseProvider &&stbc) & noexcept;`	
Parameters (in):	stbc	The SynchronizedTimeBaseProvider object to be moved.
Return value:	SynchronizedTimeBase Provider &	The moved SynchronizedTimeBaseProvider object.
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Move assignment operator for SynchronizedTimeBaseProvider.	

⌋

8.4.1.1.1.5 Destructor

[SWS_TS_01106] Definition of API function ara::tsync::SynchronizedTimeBase Provider::~SynchronizedTimeBaseProvider

Upstream requirements: RS_TS_00005

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`
Syntax:	`~SynchronizedTimeBaseProvider () noexcept;`
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Destructor for SynchronizedTimeBaseProvider.

⌋

8.4.1.1.2 Constructors

8.4.1.1.2.1 SynchronizedTimeBaseProvider

[SWS_TS_01101] Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider

Upstream requirements: RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`explicit SynchronizedTimeBaseProvider (const ara::core::InstanceSpecifier &specifier) noexcept;`	
Parameters (in):	specifier	InstanceSpecifier to an PortPrototype of a PortPrototype typed by a SynchronizedTimeBaseProviderInterface
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Violations:	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `TimeSyncPortPrototypeToTimeBaseMapping` needs to be typed by a `SynchronizedTimeBaseProviderInterface`.
Description:	SynchronizedTimeBaseProvider constructor.	

⌋

8.4.1.1.3 Member Functions

8.4.1.1.3.1 GetCurrentTime

[SWS_TS_01109] Definition of API function ara::tsync::SynchronizedTimeBase Provider::GetCurrentTime

Upstream requirements: RS_TS_00026, RS_TS_00005

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`ara::tsync::Timestamp GetCurrentTime () const noexcept;`	
Return value:	ara::tsync::Timestamp	The current time as clock specific time point.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Method to obtain the current time (regardless of the current sync status).	

⌋

8.4.1.1.3.2 GetRateDeviation

[SWS_TS_01111] Definition of API function ara::tsync::SynchronizedTimeBase Provider::GetRateDeviation

Upstream requirements: RS_TS_00018

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`double GetRateDeviation () const noexcept;`	
Return value:	double	The current rate deviation.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Method to obtain the current rate deviation of the clock.	

⌋

8.4.1.1.3.3 GetUserData

[SWS_TS_01113] Definition of API function ara::tsync::SynchronizedTimeBase Provider::GetUserData

Upstream requirements: RS_TS_00021, RS_TS_00014

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`UserData GetUserData () const noexcept;`	
Return value:	UserData	The current user data of the Time Base.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	A method to return the user defined data of the time base.	

⌋

8.4.1.1.3.4 RegisterTimeValidationNotification

[SWS_TS_01114] Definition of API function ara::tsync::SynchronizedTimeBase Provider::RegisterTimeValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`void RegisterTimeValidationNotification (ProviderTimeBaseValidation Notification &timeBaseValidationNotification) noexcept;`	
Parameters (in):	timeBaseValidation Notification	time provider application notification object that will be notified about the availability of a new data block recorded for the Time Base
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Method that can be used by time provider applications to receive time sync parameters. A maximum of one notifier can be registered. Every further registration overwrites the current registration.	

⌋

8.4.1.1.3.5 SetRateCorrection

[SWS_TS_01110] Definition of API function ara::tsync::SynchronizedTimeBase Provider::SetRateCorrection

Upstream requirements: RS_TS_00029, RS_TS_00026, RS_TS_00018

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`ara::core::Result< void > SetRateCorrection (double rateCorrection) noexcept;`	
Parameters (in):	rateCorrection	Rate correction factor applied to the local clock value. A factor smaller than 1.0 slows down the local clock by that factor, a value greater than 1.0 speeds up the clock.
Return value:	ara::core::Result< void >	-
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Errors:	TsyncErrc::kLimits Exceeded	no_rollback_semantics
		The value exceeds valid value range
Description:	This method can be used to set the rate correction that will be applied to time values.	

⌋

8.4.1.1.3.6 SetTime

[SWS_TS_01107] Definition of API function ara::tsync::SynchronizedTimeBase Provider::SetTime

Upstream requirements: RS_TS_00010, RS_TS_00026

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`ara::core::Result< void > SetTime (ara::tsync::Timestamp timePoint, const &UserData userData) noexcept;`	
Parameters (in):	timePoint	The time information to be set.
	userData	The user data to be set.
Return value:	ara::core::Result< void >	-
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Errors:	TsyncErrc::kInvalidUser DataSize	no_rollback_semantics
		The length of the user data exceeded the limit defined in UserData

▽

△

Description:	A method that can be used to set a new time value for the clock. Setting a new time also triggers transmission on the bus.

⌋

8.4.1.1.3.7 SetUserData

[SWS_TS_01112] Definition of API function ara::tsync::SynchronizedTimeBase Provider::SetUserData

Upstream requirements: RS_TS_00029, RS_TS_00026, RS_TS_00015

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`ara::core::Result< void > SetUserData (const &UserData userData) noexcept;`	
Parameters (in):	userData	The user data to be set.
Return value:	ara::core::Result< void >	-
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Errors:	TsyncErrc::kInvalidUser DataSize	no_rollback_semantics
		The length of the user data exceeded the limit defined in UserData
Description:	Method that can be used to set user data.	

⌋

8.4.1.1.3.8 UnregisterTimeValidationNotification

[SWS_TS_01115] Definition of API function ara::tsync::SynchronizedTimeBase Provider::UnregisterTimeValidationNotification

Upstream requirements: RS_TS_00034, RS_TS_00030

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`
Syntax:	`void UnregisterTimeValidationNotification () noexcept;`

▽

△

Return value:	None
Exception Safety:	exception safe
Thread Safety:	non_threadsafe
Description:	Method that can be used by time provider applications to receive time sync parameters.

⌋

8.4.1.1.3.9 UpdateTime

[SWS_TS_01108] Definition of API function ara::tsync::SynchronizedTimeBase Provider::UpdateTime

Upstream requirements: RS_TS_00010, RS_TS_00011, RS_TS_00029, RS_TS_00026, RS_-TS_00009

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_provider.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseProvider`	
Syntax:	`ara::core::Result< void > UpdateTime (ara::tsync::Timestamp timePoint, const &UserData userData) noexcept;`	
Template param:	Duration	The duration type of the time point passed as parameter.
Parameters (in):	timePoint	The time information to be set.
	userData	The user data to be set.
Return value:	ara::core::Result< void >	-
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Errors:	TsyncErrc::kInvalidUser DataSize	no_rollback_semantics
		The length of the user data exceeded the limit defined in UserData
Description:	A method that can be used to set a new time value for the clock. The clock value is only updated locally, transmission on the bus will happen in the next cycle.	

⌋

8.5 Header: ara/tsync/synchronized_time_base_status.h

8.5.1 Non-Member Types

8.5.1.1 Enumeration: LeapJump

[SWS_TS_01051] Definition of API enum ara::tsync::LeapJump

Upstream requirements: RS_TS_00009

⌈

Kind:	enumeration	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"	
Scope:	`namespace ara::tsync`	
Symbol:	LeapJump	
Underlying type:	std::uint32_t	
Syntax:	`enum class LeapJump : std::uint32_t {...};`	
Values:	kTimeLeapNone= 0	No adjustment back or greater than a certain threshold has been made.
	kTimeLeapFuture= 1	An adjustment greater than a certain threshold has been made.
	kTimeLeapPast= 2	An adjustment back in time greater than a certain threshold has been made.
Description:	Enumeration that is used to express the leap jump of a time base. .	

⌋

8.5.1.2 Enumeration: SynchronizationStatus

[SWS_TS_01050] Definition of API enum ara::tsync::SynchronizationStatus

Upstream requirements: RS_TS_00009

⌈

Kind:	enumeration	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"	
Scope:	`namespace ara::tsync`	
Symbol:	SynchronizationStatus	
Underlying type:	std::uint32_t	
Syntax:	`enum class SynchronizationStatus : std::uint32_t {...};`	
Values:	kNotSynchronizedUntil Startup= 0	The TB is not synchronized until startup (inital state)
	kTimeOut= 0x1	The TB was not synchronized within a certain time frame.

▽

△

	kSynchronized= 0x2	The TB is in sync with the time master.
	kSynchToGateway= 0x3	The TB is in sync with the gateway.
Description:	Enumeration that is used to express the communication state of a time base. .	

⌋

8.5.2 Class: SynchronizedTimeBaseStatus

[SWS_TS_01052] Definition of API class ara::tsync::SynchronizedTimeBaseStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	class
Header file:	#include "ara/tsync/synchronized_time_base_status.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	SynchronizedTimeBaseStatus
Syntax:	`class SynchronizedTimeBaseStatus final {...};`
Description:	This class represents a snapshot of a time point including his states. .

⌋

8.5.2.1 Public Member Functions

8.5.2.1.1 Special Member Functions

8.5.2.1.1.1 Move Constructor

[SWS_TS_01057] Definition of API function ara::tsync::SynchronizedTimeBaseStatus::SynchronizedTimeBaseStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_status.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`
Syntax:	`SynchronizedTimeBaseStatus (SynchronizedTimeBaseStatus &&) noexcept;`

▽

△

DIRECTION NOT DEFINED	SynchronizedTimeBase Status &&	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor of SynchronizedTimeBaseStatus.	

⌋

8.5.2.1.1.2 Copy Constructor

[SWS_TS_01058] Definition of API function ara::tsync::SynchronizedTimeBase Status::SynchronizedTimeBaseStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`SynchronizedTimeBaseStatus (const SynchronizedTimeBaseStatus &) noexcept;`	
DIRECTION NOT DEFINED	const SynchronizedTime BaseStatus &	--
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Copy constructor of SynchronizedTimeBaseStatus (needed for return by value)	

⌋

8.5.2.1.1.3 Default Constructor

[SWS_TS_01061] Definition of API function ara::tsync::SynchronizedTimeBase Status::SynchronizedTimeBaseStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`SynchronizedTimeBaseStatus ()=delete;`	

▽

△

Description:	Constructor of SynchronizedTimeBaseStatus cannot be used.

⌋

8.5.2.1.1.4 Move Assignment Operator

[SWS_TS_01059] Definition of API function ara::tsync::SynchronizedTimeBase Status::operator=

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`SynchronizedTimeBaseStatus & operator= (SynchronizedTimeBaseStatus &&) & noexcept;`	
DIRECTION NOT DEFINED	SynchronizedTimeBase Status &&	--
Return value:	SynchronizedTimeBase Status &	moved object
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Move assignment operator of SynchronizedTimeBaseStatus.	

⌋

8.5.2.1.1.5 Copy Assignment Operator

[SWS_TS_01060] Definition of API function ara::tsync::SynchronizedTimeBase Status::operator=

Upstream requirements: RS_TS_00021

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_status.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`
Syntax:	`SynchronizedTimeBaseStatus & operator= (const SynchronizedTimeBase Status &) noexcept;`

▽

△

DIRECTION NOT DEFINED	const SynchronizedTime BaseStatus &	--
Return value:	SynchronizedTimeBase Status &	copied object
Exception Safety:	exception safe	
Thread Safety:	non_threadsafe	
Description:	Copy assignment operator of SynchronizedTimeBaseStatus.	

⌋

8.5.2.1.1.6 Destructor

[SWS_TS_01062] Definition of API function ara::tsync::SynchronizedTimeBase Status::~SynchronizedTimeBaseStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	function
Header file:	#include "ara/tsync/synchronized_time_base_status.h"
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`
Syntax:	`~SynchronizedTimeBaseStatus () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Destructor of SynchronizedTimeBaseStatus.

⌋

8.5.2.1.2 Member Functions

8.5.2.1.2.1 GetCreationLocalTime

[SWS_TS_14176] Definition of API function ara::tsync::SynchronizedTimeBaseStatus::GetCreationLocalTime

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`ara::tsync::Timestamp GetCreationLocalTime () const noexcept;`	
Return value:	ara::tsync::Timestamp	The point in time at which this object was created. Time point is expressed in local clock (ara::core::SteadyClock)
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	A method to obtain the local time when this object was created.	

⌋

8.5.2.1.2.2 GetCreationTime

[SWS_TS_01055] Definition of API function ara::tsync::SynchronizedTimeBaseStatus::GetCreationTime

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`ara::core::Optional< ara::tsync::Timestamp > GetCreationTime () const noexcept;`	
Return value:	ara::core::Optional< ara::tsync::Timestamp >	The point in time at which this object was created. Time point is expressed in synchronized time. If time was never synchronized (SynchronizationStatus = kNotSynchronizedUntilStartup), the returned optional is empty (i.e., no valid value).
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	A method to obtain the creation time of this object.	

⌋

8.5.2.1.2.3 GetLeapJump

[SWS_TS_01054] Definition of API function ara::tsync::SynchronizedTimeBase Status::GetLeapJump

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`LeapJump GetLeapJump () const noexcept;`	
Return value:	LeapJump	LeapJump
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Method that can be used to determin the direction of a leap jump. Only the jump until the previous object creation is included.	

⌋

8.5.2.1.2.4 GetSynchronizationStatus

[SWS_TS_01053] Definition of API function ara::tsync::SynchronizedTimeBase Status::GetSynchronizationStatus

Upstream requirements: RS_TS_00021

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`SynchronizationStatus GetSynchronizationStatus () const noexcept;`	
Return value:	SynchronizationStatus	SynchronizationStatus
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Method that return the synchronization state the object was created.	

⌋

8.5.2.1.2.5 GetUserData

[SWS_TS_01056] Definition of API function ara::tsync::SynchronizedTimeBase Status::GetUserData

Upstream requirements: RS_TS_00021, RS_TS_00014

⌈

Kind:	function	
Header file:	#include "ara/tsync/synchronized_time_base_status.h"	
Scope:	`class ara::tsync::SynchronizedTimeBaseStatus`	
Syntax:	`ara::core::Span< const ara::core::Byte > GetUserData () const noexcept;`	
Return value:	ara::core::Span< const ara::core::Byte >	The current user data received from Time Master of the Time Base.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	A method to return the user defined data of the time base.	

⌋

8.6 Header: ara/tsync/time_precision_measurement_type.h

8.6.1 Struct: TimePrecisionMeasurement

[SWS_TS_01400] Definition of API class ara::tsync::TimePrecisionMeasurement

Upstream requirements: RS_TS_00034

⌈

Kind:	struct
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	TimePrecisionMeasurement
Syntax:	`struct TimePrecisionMeasurement final {...};`
Description:	Structure with detailed data for precision measurement of the Time Slave .

⌋

8.6.1.1 Public Member Variables

8.6.1.1.1 glbNanoSeconds

[SWS_TS_01402] Definition of API variable ara::tsync::TimePrecisionMeasurement::glbNanoSeconds

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	glbNanoSeconds
Type:	`std::uint32_t`
Syntax:	`std::uint32_t glbNanoSeconds;`
Description:	Nanoseconds of the Local Time Base directly after synchronization with the Global Time Base. Range: 0..999999999 (4 Byte)

⌋

8.6.1.1.2 glbSeconds

[SWS_TS_01401] Definition of API variable ara::tsync::TimePrecisionMeasurement::glbSeconds

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	glbSeconds
Type:	`std::uint32_t`
Syntax:	`std::uint32_t glbSeconds;`
Description:	Seconds of the Local Time Base directly after synchronization with the Global Time Base. Range: 0..4294967295 (4 Byte)

⌋

8.6.1.1.3 locNanoSeconds

[SWS_TS_01407] Definition of API variable ara::tsync::TimePrecisionMeasurement::locNanoSeconds

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	locNanoSeconds
Type:	`std::uint32_t`
Syntax:	`std::uint32_t locNanoSeconds;`
Description:	Nanoseconds of the Local Time Base directly before synchronization with the Global Time Base. Range: 0..999999999 (4 Byte)

⌋

8.6.1.1.4 locSeconds

[SWS_TS_01406] Definition of API variable ara::tsync::TimePrecisionMeasurement::locSeconds

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	locSeconds
Type:	`std::uint32_t`
Syntax:	`std::uint32_t locSeconds;`
Description:	Seconds of the Local Time Base directly before synchronization with the Global Time Base. Range: 0..4294967295 (4 Byte)

⌋

8.6.1.1.5 pathDelay

[SWS_TS_01408] Definition of API variable ara::tsync::TimePrecisionMeasurement::pathDelay

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	pathDelay
Type:	`std::uint32_t`
Syntax:	`std::uint32_t pathDelay;`
Description:	Current propagation delay in nanoseconds. Range: 0..4294967295 (4 Byte)

⌋

8.6.1.1.6 rateDeviation

[SWS_TS_01405] Definition of API variable ara::tsync::TimePrecisionMeasurement::rateDeviation

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	rateDeviation
Type:	`std::int16_t`
Syntax:	`std::int16_t rateDeviation;`
Description:	Calculated Rate Deviation directly after rate deviation measurement. Range: 0..+-32000 (2 Byte)

⌋

8.6.1.1.7 timeBaseStatus

[SWS_TS_01403] Definition of API variable ara::tsync::TimePrecisionMeasurement::timeBaseStatus

Upstream requirements: RS_TS_00034, RS_TS_00021

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	timeBaseStatus
Type:	`std::uint8_t`
Syntax:	`std::uint8_t timeBaseStatus;`
Description:	TimeBaseStatus Time Base Status of the Local Time Base directly after synchronization with the Global Time Base TIMEOUT 0x01 Bit 0 (LSB): 0x00: No Timeout on receiving Synchronisation Messages 0x01: Timeout on receiving Synchronisation Messages SYNC_TO_GATEWAY 0x04 Bit 2 0x00: Local Time Base is synchronous to Global Time Master 0x04: Local Time Base updates are based on a Time Gateway below the Global Time Master GLOBAL_TIME_BASE 0x08 Bit 3 0x00: Local Time Base is based on Local Time Base reference clock only (never synchronized with Global Time Base) 0x08: Local Time Base was at least synchronized with Global Time Base one time TIMELEAP_FUTURE 0x10 Bit 4 0x00: No leap into the future within the received time for Time Base 0x10: Leap into the future within the received time for Time Base exceeds a configured threshold TIMELEAP_PAST 0x20 Bit 5 0x00: No leap into the past within the received time for Time Base 0x20: Leap into the past within the received time for Time Base exceeds a configured threshold Description Bit 1, 6, and 7 are always 0 (reserved for future usage) Variables of this type are used to express if and how a Local Time Base is synchronized to the Global Time Master. The type is a bitfield of individual status bits, although not every combination is possible, i.e. any of the bits TIMEOUT, TIMELEAP_FUTURE, TIMELEAP_PAST and SYNC_TO_GATEWAY can only be set if the GLOBAL_TIME_BASE bit is set. .

⌋

8.6.1.1.8 virtualLocalTimeLow

[SWS_TS_01404] Definition of API variable ara::tsync::TimePrecisionMeasurement::virtualLocalTimeLow

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_precision_measurement_type.h"
Scope:	`struct ara::tsync::TimePrecisionMeasurement`
Symbol:	virtualLocalTimeLow
Type:	`std::uint32_t`
Syntax:	`std::uint32_t virtualLocalTimeLow;`

▽

△

Description:	Least significant 32 bit of the Virtual Local Time directly after synchronization with the Global Time Base. Range: 0..4294967295 (4 Byte)

⌋

8.7 Header: ara/tsync/time_validation_measurement_types.h

8.7.1 Struct: PdelayInitiatorMeasurementType

[SWS_TS_00416] Definition of API class ara::tsync::PdelayInitiatorMeasurementType

Upstream requirements: RS_TS_00034

⌈

Kind:	struct
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	PdelayInitiatorMeasurementType
Syntax:	`struct PdelayInitiatorMeasurementType final {...};`
Description:	Structure with detailed timing data for the pDelay Initiator .

⌋

8.7.1.1 Public Member Variables

8.7.1.1.1 pDelay

[SWS_TS_14166] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::pDelay

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	pDelay

▽

△

Type:	`std::uint32_t`
Syntax:	`std::uint32_t pDelay;`
Description:	currently valid pDelay value

⌋

8.7.1.1.2 referenceGlobalTimestamp

[SWS_TS_14165] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::referenceGlobalTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	referenceGlobalTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp referenceGlobalTimestamp;`
Description:	value of the local instance of the Global Time of the reference Global Time Tuple

⌋

8.7.1.1.3 referenceLocalTimestamp

[SWS_TS_14164] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::referenceLocalTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	referenceLocalTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t referenceLocalTimestamp;`
Description:	value of the Virtual Local Time of the reference Global Time Tuple

⌋

8.7.1.1.4 requestOriginTimestamp

[SWS_TS_14160] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::requestOriginTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	requestOriginTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t requestOriginTimestamp;`
Description:	egress timestamp of Pdelay_Req in Virtual Local Time

⌋

8.7.1.1.5 requestReceiptTimestamp

[SWS_TS_14162] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::requestReceiptTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	requestReceiptTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp requestReceiptTimestamp;`
Description:	ingress timestamp of Pdelay_Req in Global Time taken from the received Pdelay_Resp

⌋

8.7.1.1.6 responseOriginTimestamp

[SWS_TS_14163] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::responseOriginTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	responseOriginTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp responseOriginTimestamp;`
Description:	egress timestamp of Pdelay_Resp in Global Time taken from the received Pdelay_Resp_Follow_Up

⌋

8.7.1.1.7 responseReceiptTimestamp

[SWS_TS_14161] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::responseReceiptTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	responseReceiptTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t responseReceiptTimestamp;`
Description:	ingress timestamp of Pdelay_Resp in Virtual Local Time

⌋

8.7.1.1.8 sequenceId

[SWS_TS_14167] Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::sequenceId

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayInitiatorMeasurementType`
Symbol:	sequenceId
Type:	`std::uint16_t`
Syntax:	`std::uint16_t sequenceId;`
Description:	sequence Id of sent Pdelay_Req frame

⌋

8.7.2 Struct: PdelayResponderMeasurementType

[SWS_TS_00417] Definition of API class ara::tsync::PdelayResponderMeasurementType

Upstream requirements: RS_TS_00034

⌈

Kind:	struct
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	PdelayResponderMeasurementType
Syntax:	`struct PdelayResponderMeasurementType final {...};`
Description:	Structure with detailed timing data for the pDelay Responder .

⌋

8.7.2.1 Public Member Variables

8.7.2.1.1 referenceGlobalTimestamp

[SWS_TS_14173] Definition of API variable ara::tsync::PdelayResponderMeasurementType::referenceGlobalTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayResponderMeasurementType`
Symbol:	referenceGlobalTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp referenceGlobalTimestamp;`
Description:	value of the local instance of the Global Time of the reference Global Time Tuple used to convert requestReceiptTimestamp and responseOriginTimestamp into Global Time

⌋

8.7.2.1.2 referenceLocalTimestamp

[SWS_TS_14172] Definition of API variable ara::tsync::PdelayResponderMeasurementType::referenceLocalTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayResponderMeasurementType`
Symbol:	referenceLocalTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t referenceLocalTimestamp;`
Description:	value of the Virtual Local Time of the reference Global Time Tuple used to convert request ReceiptTimestamp and responseOriginTimestamp into Global Time

⌋

8.7.2.1.3 requestReceiptTimestamp

[SWS_TS_14170] Definition of API variable ara::tsync::PdelayResponderMeasurementType::requestReceiptTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayResponderMeasurementType`
Symbol:	requestReceiptTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t requestReceiptTimestamp;`
Description:	ingress timestamp of Pdelay_Req converted to Virtual Local Time

⌋

8.7.2.1.4 responseOriginTimestamp

[SWS_TS_14171] Definition of API variable ara::tsync::PdelayResponderMeasurementType::responseOriginTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayResponderMeasurementType`
Symbol:	responseOriginTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t responseOriginTimestamp;`
Description:	egress timestamp of Pdelay_Resp converted to Virtual Local Time

⌋

8.7.2.1.5 sequenceId

[SWS_TS_14174] Definition of API variable ara::tsync::PdelayResponderMeasurementType::sequenceId

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::PdelayResponderMeasurementType`
Symbol:	sequenceId
Type:	`std::uint16_t`
Syntax:	`std::uint16_t sequenceId;`
Description:	sequence Id of received Pdelay_Req frame

⌋

8.7.3 Struct: TimeMasterMeasurementType

[SWS_TS_00414] Definition of API class ara::tsync::TimeMasterMeasurementType

Upstream requirements: RS_TS_00034

⌈

Kind:	struct
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	TimeMasterMeasurementType
Syntax:	`struct TimeMasterMeasurementType final {...};`
Description:	Structure with detailed data for validation of the Time Master .

⌋

8.7.3.1 Public Member Variables

8.7.3.1.1 preciseOriginTimestamp

[SWS_TS_14140] Definition of API variable ara::tsync::TimeMasterMeasurementType::preciseOriginTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeMasterMeasurementType`
Symbol:	preciseOriginTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp preciseOriginTimestamp;`
Description:	egress timestamp of Sync frame in Global Time

⌋

8.7.3.1.2 sequenceId

[SWS_TS_14142] Definition of API variable ara::tsync::TimeMasterMeasurementType::sequenceId

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeMasterMeasurementType`
Symbol:	sequenceId
Type:	`std::uint16_t`
Syntax:	`std::uint16_t sequenceId;`
Description:	sequence Id of sent Ethernet frame

⌋

8.7.3.1.3 syncEgressTimestamp

[SWS_TS_14141] Definition of API variable ara::tsync::TimeMasterMeasurementType::syncEgressTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeMasterMeasurementType`
Symbol:	syncEgressTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t syncEgressTimestamp;`
Description:	egress timestamp of Sync frame

⌋

8.7.4 Struct: TimeSlaveMeasurementType

[SWS_TS_00415] Definition of API class ara::tsync::TimeSlaveMeasurementType

Upstream requirements: RS_TS_00034

⌈

Kind:	struct
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	TimeSlaveMeasurementType
Syntax:	`struct TimeSlaveMeasurementType final {...};`
Description:	Structure with detailed data for validation of the Time Slave .

⌋

8.7.4.1 Public Member Variables

8.7.4.1.1 correctionField

[SWS_TS_14153] Definition of API variable ara::tsync::TimeSlaveMeasurementType::correctionField

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	correctionField
Type:	`std::int64_t`
Syntax:	`std::int64_t correctionField;`
Description:	correctionField taken from the received Follow_Up frame

⌋

8.7.4.1.2 pDelay

[SWS_TS_14155] Definition of API variable ara::tsync::TimeSlaveMeasurementType::pDelay

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	pDelay
Type:	`std::uint32_t`
Syntax:	`std::uint32_t pDelay;`
Description:	currently valid pDelay value

⌋

8.7.4.1.3 preciseOriginTimestamp

[SWS_TS_14150] Definition of API variable ara::tsync::TimeSlaveMeasurementType::preciseOriginTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	preciseOriginTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp preciseOriginTimestamp;`
Description:	preciseOriginTimestamp taken from the received Follow_Up frame

⌋

8.7.4.1.4 referenceGlobalTimestamp

[SWS_TS_14151] Definition of API variable ara::tsync::TimeSlaveMeasurementType::referenceGlobalTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	referenceGlobalTimestamp
Type:	`ara::tsync::Timestamp`
Syntax:	`ara::tsync::Timestamp referenceGlobalTimestamp;`
Description:	SyncLocal Time Tuple (Global Time part) .

⌋

8.7.4.1.5 referenceLocalTimestamp

[SWS_TS_14154] Definition of API variable ara::tsync::TimeSlaveMeasurementType::referenceLocalTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	referenceLocalTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t referenceLocalTimestamp;`
Description:	SyncLocal Time Tuple (Virtual Local Time part) .

⌋

8.7.4.1.6 sequenceId

[SWS_TS_14156] Definition of API variable ara::tsync::TimeSlaveMeasurementType::sequenceId

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	sequenceId
Type:	`std::uint16_t`
Syntax:	`std::uint16_t sequenceId;`
Description:	sequence Id of received Sync frame

⌋

8.7.4.1.7 syncIngressTimestamp

[SWS_TS_14152] Definition of API variable ara::tsync::TimeSlaveMeasurementType::syncIngressTimestamp

Upstream requirements: RS_TS_00034

⌈

Kind:	variable
Header file:	#include "ara/tsync/time_validation_measurement_types.h"
Scope:	`struct ara::tsync::TimeSlaveMeasurementType`
Symbol:	syncIngressTimestamp
Type:	`std::uint64_t`
Syntax:	`std::uint64_t syncIngressTimestamp;`
Description:	ingress timestamp of Sync frame converted to Virtual Local Time

⌋

8.8 Header: ara/tsync/timestamp.h

8.8.1 Non-Member Types

8.8.1.1 Type Alias: Timestamp

[SWS_TS_01251] Definition of API type ara::tsync::Timestamp

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	type alias
Header file:	#include "ara/tsync/timestamp.h"
Scope:	`namespace ara::tsync`
Symbol:	Timestamp
Syntax:	`using Timestamp = std::chrono::time_point<TimeBase, std::chrono::nanoseconds>;`
Description:	Standard timestamp type as alias of a generic time_point .

⌋

8.8.2 Struct: TimeBase

[SWS_TS_01260] Definition of API class ara::tsync::TimeBase

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	struct
Header file:	#include "ara/tsync/timestamp.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	TimeBase
Syntax:	`struct TimeBase {...};`
Description:	The class TimeBase is a pseudo-clock. The class TimeBase is a pseudo-clock that is used as the first template argument to std::chrono::time_point to indicate that the time point represents local time with respect of a not-yet-specified timeBaseResource.

⌋

8.8.2.1 Public Member Types

8.8.2.1.1 Type Alias: duration

[SWS_TS_01263] Definition of API type ara::tsync::TimeBase::duration

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	type alias
Header file:	#include "ara/tsync/timestamp.h"
Scope:	`struct ara::tsync::TimeBase`
Symbol:	duration
Syntax:	`using duration = std::chrono::duration<rep, period>;`
Description:	required type declaration to fulfill the C++ Clock requirements used to measure the time since epoch

⌋

8.8.2.1.2 Type Alias: period

[SWS_TS_01262] Definition of API type ara::tsync::TimeBase::period

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	type alias
Header file:	#include "ara/tsync/timestamp.h"
Scope:	`struct ara::tsync::TimeBase`
Symbol:	period
Syntax:	`using period = std::nano;`
Description:	required type declaration to fulfill the C++ Clock requirements representing the tick period of the duration

⌋

8.8.2.1.3 Type Alias: rep

[SWS_TS_01261] Definition of API type ara::tsync::TimeBase::rep

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	type alias
Header file:	#include "ara/tsync/timestamp.h"
Scope:	`struct ara::tsync::TimeBase`
Symbol:	rep
Syntax:	`using rep = std::int64_t;`
Description:	required type declaration to fulfill the C++ Clock requirements representing the number of ticks

⌋

8.8.2.1.4 Type Alias: time_point

[SWS_TS_01264] Definition of API type ara::tsync::TimeBase::time_point

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	type alias
Header file:	#include "ara/tsync/timestamp.h"
Scope:	`struct ara::tsync::TimeBase`
Symbol:	time_point
Syntax:	`using time_point = std::chrono::time_point<TimeBase>;`
Description:	represents a point in time

⌋

8.8.2.2 Public Member Variables

8.8.2.2.1 is_steady

[SWS_TS_01265] Definition of API variable ara::tsync::TimeBase::is_steady

Upstream requirements: RS_AP_00130, RS_TS_00005

⌈

Kind:	variable
Header file:	#include "ara/tsync/timestamp.h"
Scope:	`struct ara::tsync::TimeBase`
Symbol:	is_steady
Type:	`bool`
Syntax:	`static constexpr bool is_steady = false;`
Description:	required constant to fulfill the C++ Clock requirements as steady clock

⌋

8.9 Header: ara/tsync/tsync_error_domain.h

8.9.1 Non-Member Types

8.9.1.1 Enumeration: TsyncErrc

[SWS_TS_00901] Definition of API enum ara::tsync::TsyncErrc

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127, RS_AP_00154

⌈

Kind:	enumeration	
Header file:	#include "ara/tsync/tsync_error_domain.h"	
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"	
Scope:	`namespace ara::tsync`	
Symbol:	TsyncErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class TsyncErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kDaemonConnection Lost= 1	The action cannot be executed, because the connection to time sync daemon is currently lost.
	kLimitsExceeded= 2	The value exceeds valid value range
	kFunctionNotSupported= 3	The requested function is not supported
	kInvalidUserDataSize= 4	The length of the user data exceeded the limit defined in UserData
Description:	Defines an enumeration class for the Time Synchronization error codes.	

⌋

8.9.2 Non-Member Functions

8.9.2.1 Other

8.9.2.1.1 GetTsyncErrorDomain

[SWS_TS_00909] Definition of API function ara::tsync::GetTsyncErrorDomain

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00154

⌈

Kind:	function
Header file:	#include "ara/tsync/tsync_error_domain.h"
Scope:	`namespace ara::tsync`
Syntax:	`const ara::core::ErrorDomain & GetTsyncErrorDomain () noexcept;`

▽

△

Return value:	const ara::core::Error Domain &	Return a reference to the global TsyncErrorDomain object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns a reference to the global TsyncErrorDomain object.	

⌋

8.9.2.1.2 MakeErrorCode

[SWS_TS_00910] Definition of API function ara::tsync::MakeErrorCode

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130, RS_AP_00154

⌈

Kind:	function	
Header file:	#include "ara/tsync/tsync_error_domain.h"	
Scope:	`namespace ara::tsync`	
Syntax:	`ara::core::ErrorCode MakeErrorCode (ara::tsync::TsyncErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	Error code number.
	data	Vendor defined data associated with the error.
Return value:	ara::core::ErrorCode	An ErrorCode object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Creates an instance of ErrorCode.	

⌋

8.9.3 Class: TsyncErrorDomain

[SWS_TS_00904] Definition of API class ara::tsync::TsyncErrorDomain

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127, RS_AP_00154, RS_AP_00150

⌈

Kind:	class
Header file:	#include "ara/tsync/tsync_error_domain.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`

▽

△

Symbol:	TsyncErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class TsyncErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::tsync::TsyncErrorDomain` in [SWS_CORE_90023]
Description:	Defines a class representing the Time Synchronization error domain.

⌋

8.9.3.1 Public Member Types

8.9.3.1.1 Type Alias: Errc

[SWS_TS_01266] Definition of API type ara::tsync::TsyncErrorDomain::Errc

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	type alias
Header file:	#include "ara/tsync/tsync_error_domain.h"
Scope:	`class ara::tsync::TsyncErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = TsyncErrc;`
Description:	Alias for the error code value enumeration

⌋

8.9.3.1.2 Type Alias: Exception

[SWS_TS_01267] Definition of API type ara::tsync::TsyncErrorDomain::Exception

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	type alias
Header file:	#include "ara/tsync/tsync_error_domain.h"
Scope:	`class ara::tsync::TsyncErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = TsyncException;`
Description:	Alias for the exception base class

⌋

8.9.3.2 Public Member Functions

8.9.3.2.1 Special Member Functions

8.9.3.2.1.1 Default Constructor

[SWS_TS_00905] Definition of API function ara::tsync::TsyncErrorDomain::TsyncErrorDomain

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	function
Header file:	#include "ara/tsync/tsync_error_domain.h"
Scope:	`class ara::tsync::TsyncErrorDomain`
Syntax:	`TsyncErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructs a new TsyncErrorDomain object.

⌋

8.9.3.2.2 Member Functions

8.9.3.2.2.1 Message

[SWS_TS_00907] Definition of API function ara::tsync::TsyncErrorDomain::Message

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/tsync/tsync_error_domain.h"	
Scope:	`class ara::tsync::TsyncErrorDomain`	
Syntax:	`const char * Message (CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	The error code number.
Return value:	const char *	The message associated with the error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the message associated with errorCode.	

⌋

8.9.3.2.2.2 Name

[SWS_TS_00906] Definition of API function ara::tsync::TsyncErrorDomain::Name

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/tsync/tsync_error_domain.h"	
Scope:	`class ara::tsync::TsyncErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"Tsync"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns a string constant associated with TsyncErrorDomain.	

⌋

8.9.3.2.2.3 ThrowAsException

[SWS_TS_00908] Definition of API function ara::tsync::TsyncErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/tsync/tsync_error_domain.h"	
Scope:	`class ara::tsync::TsyncErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	The error to throw.
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Creates a new instance of TsyncException from errorCode and throws it as a C++ exception. As per SWS_CORE_10304 this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.9.4 Class: TsyncException

[SWS_TS_00902] Definition of API class ara::tsync::TsyncException

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127, RS_AP_00154, RS_AP_00150

⌈

Kind:	class
Header file:	#include "ara/tsync/tsync_error_domain.h"
Forwarding header file:	#include "ara/tsync/tsync_fwd.h"
Scope:	`namespace ara::tsync`
Symbol:	TsyncException
Base class:	ara::core::Exception
Syntax:	`class TsyncException : public ara::core::Exception {...};`
Description:	Defines a class for exceptions to be thrown by the Time Synchronization.

⌋

8.9.4.1 Public Member Functions

8.9.4.1.1 Constructors

8.9.4.1.1.1 TsyncException

[SWS_TS_00903] Definition of API function ara::tsync::TsyncException::TsyncException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/tsync/tsync_error_domain.h"	
Scope:	`class ara::tsync::TsyncException`	
Syntax:	`explicit TsyncException (ara::core::ErrorCode errorCode) noexcept;`	
Parameters (in):	errorCode	The error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructs a new TsyncException object containing an error code.	

⌋

9 Service Interfaces

This functional cluster does not define any provided or required service interfaces.

10 Configuration

The configuration model of this functional cluster is defined in [11]. This chapter defines the default values for attributes and semantic constraints for elements specified in [11] that are part of the configuration model of this functional cluster.

10.1 Default Values

This functional cluster does not define any default values for attributes specified in [11].

10.2 Semantic Constraints

[SWS_TS_CONSTR_00001] Configurable Namespace for TimeSynchronization ⌈Configurable Namespace for TimeSynchronization `AbstractSynchronizedTimeBaseInterface.namespace` shall never exist.⌋

A Mentioned Manifest Elements

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

This chapter is generated.

Class	***AbstractSynchronizedTimeBaseInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Time Synchronization.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumerInterface, SynchronizedTimeBaseProviderInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.1: AbstractSynchronizedTimeBaseInterface

Class	**GlobalTimeDomain**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the ability to define a global time domain. **Tags:** atp.recommendedPackage=GlobalTimeDomains			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
debounceTime	TimeValue	0..1	attr	Defines the minimum amount of time between two time sync messages are transmitted.
domainId	PositiveInteger	0..1	attr	This represents the ID of the GlobalTimeDomain used in the network messages sent on behalf of global time management.
gateway	GlobalTimeGateway	*	aggr	A GlobalTimeGateway may exist in the context of a GlobalTimeDomain to actively update the global time information as it is routed from one GlobalTimeDomain to another. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=gateway.shortName, gateway.variation Point.shortLabel vh.latestBindingTime=postBuild
globalTime CorrectionProps	GlobalTimeCorrection Props	0..1	aggr	Defintion of attributes for rate and offset correction.
globalTime Domain Property	AbstractGlobalTime DomainProps	0..1	aggr	Additional properties of the GlobalTimeDomain. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeDomainProperty, globalTime DomainProperty.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**GlobalTimeDomain**			
globalTime Master	GlobalTimeMaster	0..1	aggr	This represents the single master of a GlobalTime Domain. A GlobalTimeDomain may have no GlobalTime Domain.master, e.g. when it gets its time from a GPS receiver. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeMaster.shortName, globalTime Master.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTimeSub Domain	GlobalTimeDomain	*	ref	By this means it is possible to create a hierarchy of sub Domains where one global time domain can declare one or more other global time domains as its subDomains. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeSubDomain.globalTimeDomain, globalTimeSubDomain.variationPoint.shortLabel vh.latestBindingTime=postBuild
icvFreshness ValueId	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value for the Integrity Check Value (ICV) calculation and verification.
icvSecureCom Props	SecOcSecureCom Props	0..1	ref	Reference to a SecureComProps definition to be used for the Integrity Check Value (ICV) calculation and verification. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=icvSecureComProps.secOcSecureCom Props, icvSecureComProps.variationPoint.shortLabel vh.latestBindingTime=postBuild
maxProgression Mismatch Threshold	TimeValue	0..1	attr	This attribute defines the maximum allowed difference between local time and fallback time of the time base in seconds.
network SegmentId	NetworkSegment Identification	0..1	aggr	Defines the numerical identification of a GlobalTime sub domain.
pduTriggering	PduTriggering	0..1	ref	This PduTriggering will be taken to transmit the global time information from a GlobalTimeMaster to a the associated GlobalTimeSlaves. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pduTriggering.pduTriggering, pdu Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
slave	GlobalTimeSlave	*	aggr	This represents the collections of slaves of the Global TimeDomain. A GlobalTimeDomain may have no Global TimeDomain.slaves, e.g. when it propagates its time directly to sub domains. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=slave.shortName, slave.variationPoint.short Label vh.latestBindingTime=postBuild
syncLoss Timeout	TimeValue	0..1	attr	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain.

Table A.2: GlobalTimeDomain

Class	***GlobalTimeMaster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time master.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanMaster, GlobalTimeEthMaster, GlobalTimeFrMaster, UserDefinedGlobalTimeMaster			
Aggregated by	GlobalTimeDomain.globalTimeMaster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	The GlobalTimeMaster is bound to the Communication Connector.
immediate ResumeTime	TimeValue	0..1	attr	Defines the minimum time between an "immediate" message and the next periodic message.
isSystemWide GlobalTime Master	Boolean	0..1	attr	If set to TRUE, the GlobalTimeMaster is supposed to act as the root of global time information.
syncPeriod	TimeValue	0..1	attr	This represents the period. Unit: seconds

Table A.3: GlobalTimeMaster

Class	***GlobalTimeSlave*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time slave.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanSlave, GlobalTimeEthSlave, GlobalTimeFrSlave, UserDefinedGlobalTimeSlave			
Aggregated by	GlobalTimeDomain.slave			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	The GlobalTimeSlave is bound to the Communication Connector.
followUp TimeoutValue	TimeValue	0..1	attr	Rx timeout for the follow-up message.
timeLeapFuture Threshold	TimeValue	0..1	attr	Defines the maximum allowed positive difference between the current Local Time Base value and a newly received Global Time Base value.
timeLeap HealingCounter	PositiveInteger	0..1	attr	Defines the required number of updates to the Time Base where the time difference to the previous received value has to remain within the bounds of timeLeapFuture Threshold and timeLeapPastThreshold until that Time Base is considered healed.
timeLeapPast Threshold	TimeValue	0..1	attr	Defines the maximum allowed negative difference between the current Local Time Base value and a newly received Global Time Base value.

Table A.4: GlobalTimeSlave

Class	***PortInterface*** (abstract)
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	Abstract base class for an interface that is either provided or required by a port of a software component.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	***PortInterface*** (abstract)			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSynchronizedTimeBaseInterface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallStateSwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, NetworkManagementPortInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=namespace.shortName

Table A.5: PortInterface

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table A.6: PortPrototype

Class	SynchronizedTimeBaseConsumerInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Consumer. **Tags:** atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.7: SynchronizedTimeBaseConsumerInterface

Class	SynchronizedTimeBaseProviderInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Provider. **Tags:** atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.8: SynchronizedTimeBaseProviderInterface

Class	TimeBaseProviderToPersistencyMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the ability to define a mapping between a TimeBaseProvider and a PersistencyDeploymentElement for the purpose of storing and retrieving the time value. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cyclicBackupInterval	TimeValue	0..1	attr	Time interval in seconds to store the time base value periodically to persistence.
persistencyDeploymentElement	PersistencyDeploymentElement	0..1	ref	This reference represents the PersistencyDeploymentElement where the time value shall be stored in and retrieved from.
timeBaseProvider	SynchronizedTimeBaseProvider	0..1	ref	This reference represents the mapped TimeBaseProvider.

Table A.9: TimeBaseProviderToPersistencyMapping

Class	**TimeSyncCorrection**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes used for the correction of time synchronization.			
Base	*ARObject*			
Aggregated by	SynchronizedTimeBaseProvider.timeSyncCorrection			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowProvider RateCorrection	Boolean	0..1	attr	Defines whether the rate correction value of a Time Base can be set by means of the method setRateCorrection(). false: rate correction cannot be set by method setRate Correction(). true: rate correction can be set by method setRate Correction().
offsetCorrection AdaptionInterval	TimeValue	0..1	attr	Defines the interval during which the adaptive rate correction cancels out the rate and time deviation. Unit: seconds.
offsetCorrection JumpThreshold	TimeValue	0..1	attr	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal and greater than this value will be corrected by immediately setting the correct time and rate in form of a jump. Unit: seconds.
providerRate DeviationMax	PositiveInteger	0..1	attr	This attribute describes the maximum allowed absolute value of the rate deviation value [unit: ppm].
rateCorrections Per Measurement Duration	PositiveInteger	0..1	attr	Number of simultaneous rate measurements to determine the current rate deviation.
rateDeviation Measurement Duration	TimeValue	0..1	attr	Time span used to calculate the rate deviation. Unit: seconds.

Table A.10: TimeSyncCorrection

Class	**TimeSyncPortPrototypeToTimeBaseMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class provides the ability to map a PortPrototype typed by a AbstractSynchronizedTimeBase Interface to a TimeBaseResource in the context of a Process. **Tags:** atp.recommendedPackage=TimeSyncPortPrototypeToTimeBaseMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the context Process this mapping applies to.
timeBase Resource	TimeBaseResource	0..1	ref	Reference to the mapped TimeBaseResource.
timeSyncPPort Prototype	PPortPrototype	0..1	iref	Instance reference to the mapped PPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
timeSyncRPort Prototype	RPortPrototype	0..1	iref	Instance reference to the mapped RPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table A.11: TimeSyncPortPrototypeToTimeBaseMapping

B Demands and constraints on Base Software (normative)

This functional cluster defines no demands or constraints for the Base Software on which the AUTOSAR Adaptive Platform is running on (usually a POSIX-compatible operating system).

C Interfaces to other Functional Clusters (informative)

This chapter provides a reference of the Platform Extension Interfaces defined by this functional cluster. Platform Extension Interfaces are intended to be used/provided by an OEM or Integrator to extend the functionality of the AUTOSAR Adaptive Platform.

C.1 Header: apext/tsync/fvm.h

C.1.1 Struct: FVContainer

[SWS_TS_14194] Definition of API class apext::tsync::FVContainer

Upstream requirements: RS_TS_20072

⌈

Kind:	struct
Header file:	#include "apext/tsync/fvm.h"
Forwarding header file:	#include "apext/tsync/tsync_fwd.h"
Scope:	`namespace apext::tsync`
Symbol:	FVContainer
Syntax:	`struct FVContainer {...};`
Description:	A freshness value container to hold the length of freshness value in bits and the freshness value itself as vector.

⌋

C.1.1.1 Public Member Variables

C.1.1.1.1 length

[SWS_TS_14195] Definition of API variable apext::tsync::FVContainer::length

Upstream requirements: RS_TS_20072

⌈

Kind:	variable
Header file:	#include "apext/tsync/fvm.h"
Scope:	`struct apext::tsync::FVContainer`
Symbol:	length
Type:	`std::uint64_t`
Syntax:	`std::uint64_t length;`

▽

△

Description:	Length in bits of the freshness value passed in `apext::tsync::FVContainer`.

⌋

C.1.1.1.2 value

[SWS_TS_14196] Definition of API variable apext::tsync::FVContainer::value

Upstream requirements: RS_TS_20072

⌈

Kind:	variable
Header file:	#include "apext/tsync/fvm.h"
Scope:	`struct apext::tsync::FVContainer`
Symbol:	value
Type:	`ara::core::Vector< std::uint8_t >`
Syntax:	`ara::core::Vector<std::uint8_t> value;`
Description:	Vector of bytes containing the freshness value. Depending on whether the container is used as an input or returning value by the method it will contain either the full freshness or truncated values.

⌋

C.1.2 Class: FVM

[SWS_TS_14197] Definition of API class apext::tsync::FVM

Upstream requirements: RS_TS_20072

⌈

Kind:	class
Header file:	#include "apext/tsync/fvm.h"
Forwarding header file:	#include "apext/tsync/tsync_fwd.h"
Scope:	`namespace apext::tsync`
Symbol:	FVM
Syntax:	`class FVM {...};`
Description:	A freshness value management interface for ara::tsync.

⌋

C.1.2.1 Public Member Functions

C.1.2.1.1 Member Functions

C.1.2.1.1.1 GetRxFreshness

[SWS_TS_14198] Definition of API function apext::tsync::FVM::GetRxFreshness

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm.h"	
Scope:	`class apext::tsync::FVM`	
Syntax:	`ara::core::Result< FVContainer > GetRxFreshness (std::uint16_t FreshnessValueID, const FVContainer &TruncatedFreshnessValue, std::uint16_t AuthVerifyAttempts) noexcept;`	
Parameters (in):	FreshnessValueID	the identifier of the freshness value.
	TruncatedFreshness Value	the `apext::tsync::FVContainer` with the values from the received Secured message.
	AuthVerifyAttempts	the number of authentication verify attempts of this message since the last reception. The value is `0` for the first attempt and incremented on every unsuccessful verification attempt.
Return value:	ara::core::Result< FVContainer >	an `ara::core::Result` containing a `ara::core::Result::value_type` containing a `apext::tsync::FVContainer` that holds the freshness value to be used for the calculation of the authenticator by TimeSync, or recoverable error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	FvmErrc::kFVNot Available	rollback_semantics
		the Freshness Value is not available.
Description:	Used by TimeSync to obtain the current freshness value for received messages.	

⌋

C.1.2.1.1.2 GetTxFreshness

[SWS_TS_14199] Definition of API function apext::tsync::FVM::GetTxFreshness

Upstream requirements: RS_TS_20072

⌈

Kind:	function
Header file:	#include "apext/tsync/fvm.h"
Scope:	`class apext::tsync::FVM`

▽

△

Syntax:	`ara::core::Result< FVContainer > GetTxFreshness (std::uint16_t FreshnessValueID) noexcept;`	
Parameters (in):	FreshnessValueID	the identifier of the freshness value.
Return value:	ara::core::Result< FVContainer >	an `ara::core::Result` containing a `ara::core::Result::value_type` containing a `apext::tsync::FVContainer` that holds the freshness value to be used for the calculation of the authenticator by TimeSync, or recoverable error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	FvmErrc::kFVNot Available	rollback_semantics
		the Freshness Value is not available.
Description:	Used by TimeSync to obtain the current freshness value for transmitted messages.	

⌋

C.1.2.1.1.3 Initialize

[SWS_TS_14200] Definition of API function apext::tsync::FVM::Initialize

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm.h"	
Scope:	`class apext::tsync::FVM`	
Syntax:	`ara::core::Result< void > Initialize () noexcept;`	
Return value:	ara::core::Result< void >	an `ara::core::Result` containing a `ara::core::Result::value_type` containing a `void`, kFVInitializeFailed otherwise.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	FvmErrc::kFVInitialize Failed	rollback_semantics
		Initialization fails
Description:	Initializes FVM	

⌋

C.2 Header: apext/tsync/fvm_error_domain.h

C.2.1 Non-Member Types

C.2.1.1 Enumeration: FvmErrc

[SWS_TS_14202] Definition of API enum apext::tsync::FvmErrc

Upstream requirements: RS_TS_20072

⌈

Kind:	enumeration	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Forwarding header file:	#include "apext/tsync/tsync_fwd.h"	
Scope:	`namespace apext::tsync`	
Symbol:	FvmErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class FvmErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kFVNotAvailable= 1	Recoverable Error meaning the Freshness Value is not available.
	kFVInitializeFailed= 2	Unrecoverable Error meaning the Freshness Value Manager could not be used.
Description:	The enumeration class defines the error codes for the `apext::tsync::FvmErrorDomain`.	

⌋

C.2.2 Non-Member Functions

C.2.2.1 Other

C.2.2.1.1 GetFvmErrorDomain

[SWS_TS_14212] Definition of API function apext::tsync::GetFvmErrorDomain

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Scope:	`namespace apext::tsync`	
Syntax:	`constexpr const ara::core::ErrorDomain & GetFvmErrorDomain () noexcept;`	
Return value:	const ara::core::Error Domain &	A reference to the global `apext::tsync::FvmErrorDomain` object.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe
Description:	Returns a reference to the global `apext::tsync::FvmErrorDomain` object.

⌋

C.2.2.1.2 MakeErrorCode

[SWS_TS_14213] Definition of API function apext::tsync::MakeErrorCode

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Scope:	`namespace apext::tsync`	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (FvmErrc code, ara::core::ErrorDomain::SupportDataType data) noexcept;`	
Parameters (in):	code	The error to throw.
	data	Vendor defined data associated with the error.
Return value:	ara::core::ErrorCode	An `ara::core::ErrorCode` object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Creates an instance of `ara::core::ErrorCode`.	

⌋

C.2.3 Class: FvmErrorDomain

[SWS_TS_14205] Definition of API class apext::tsync::FvmErrorDomain

Upstream requirements: RS_TS_20072

⌈

Kind:	class
Header file:	#include "apext/tsync/fvm_error_domain.h"
Forwarding header file:	#include "apext/tsync/tsync_fwd.h"
Scope:	`namespace apext::tsync`
Symbol:	FvmErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class FvmErrorDomain : public ara::core::ErrorDomain {...};`
Description:	Defines a class representing the Freshness Value Manager error domain.

⌋

C.2.3.1 Public Member Types

C.2.3.1.1 Type Alias: Errc

[SWS_TS_14206] Definition of API type apext::tsync::FvmErrorDomain::Errc

Upstream requirements: RS_TS_20072

⌈

Kind:	type alias
Header file:	#include "apext/tsync/fvm_error_domain.h"
Scope:	`class apext::tsync::FvmErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = FvmErrc;`
Description:	Alias for the error code value enumeration.

⌋

C.2.3.1.2 Type Alias: Exception

[SWS_TS_14207] Definition of API type apext::tsync::FvmErrorDomain::Exception

Upstream requirements: RS_TS_20072

⌈

Kind:	type alias
Header file:	#include "apext/tsync/fvm_error_domain.h"
Scope:	`class apext::tsync::FvmErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = FvmException;`
Description:	Alias for the exception base class.

⌋

C.2.3.2 Public Member Functions

C.2.3.2.1 Special Member Functions

C.2.3.2.1.1 Default Constructor

[SWS_TS_14208] Definition of API function apext::tsync::FvmErrorDomain::FvmErrorDomain

Upstream requirements: RS_TS_20072

⌈

Kind:	function
Header file:	#include "apext/tsync/fvm_error_domain.h"
Scope:	`class apext::tsync::FvmErrorDomain`
Syntax:	`constexpr FvmErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Constructs a new `apext::tsync::FvmErrorDomain` object.

⌋

C.2.3.2.1.2 Destructor

[SWS_TS_14201] Definition of API function apext::tsync::FvmErrorDomain::~FvmErrorDomain

Upstream requirements: RS_TS_20072

⌈

Kind:	function
Header file:	#include "apext/tsync/fvm_error_domain.h"
Scope:	`class apext::tsync::FvmErrorDomain`
Syntax:	`virtual ~FvmErrorDomain () noexcept=default;`
Exception Safety:	exception safe
Thread Safety:	implementation defined
Description:	Destructor of `apext::tsync::FvmErrorDomain`.

⌋

C.2.3.2.2 Member Functions

C.2.3.2.2.1 Message

[SWS_TS_14210] Definition of API function apext::tsync::FvmErrorDomain::Message

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Scope:	`class apext::tsync::FvmErrorDomain`	
Syntax:	`const char * Message (CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	The error code number.
Return value:	const char *	The message associated with the `errorCode`.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the message associated with the `errorCode`.	

⌋

C.2.3.2.2.2 Name

[SWS_TS_14209] Definition of API function apext::tsync::FvmErrorDomain::Name

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Scope:	`class apext::tsync::FvmErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	"Fvm"
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns a string constant associated with the `apext::tsync::FvmErrorDomain`.	

⌋

C.2.3.2.2.3 ThrowAsException

[SWS_TS_14211] Definition of API function apext::tsync::FvmErrorDomain::ThrowAsException

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Scope:	`class apext::tsync::FvmErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	The error to throw.
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Creates a new instance of `apext::tsync::FvmException` from `errorCode` and throws it. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

C.2.4 Class: FvmException

[SWS_TS_14203] Definition of API class apext::tsync::FvmException

Upstream requirements: RS_TS_20072

⌈

Kind:	class
Header file:	#include "apext/tsync/fvm_error_domain.h"
Forwarding header file:	#include "apext/tsync/tsync_fwd.h"
Scope:	`namespace apext::tsync`
Symbol:	FvmException
Base class:	ara::core::Exception
Syntax:	`class FvmException : public ara::core::Exception {...};`
Description:	Defines a class for exceptions to be thrown by the Freshness Value Manager.

⌋

C.2.4.1 Public Member Functions

C.2.4.1.1 Constructors

C.2.4.1.1.1 FvmException

[SWS_TS_14204] Definition of API function apext::tsync::FvmException::FvmException

Upstream requirements: RS_TS_20072

⌈

Kind:	function	
Header file:	#include "apext/tsync/fvm_error_domain.h"	
Scope:	`class apext::tsync::FvmException`	
Syntax:	`explicit FvmException (ara::core::ErrorCode errorCode) noexcept;`	
Parameters (in):	errorCode	The error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructs a new `apext::tsync::FvmException` object containing an error code.	

⌋

D Change History

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

D.1 Change History of this document according to AUTOSAR Release R23-11

D.1.1 Added Specification Items in R23-11

Number	Heading
[SWS_TS_00901]	Definition of API enum ara::tsync::TsyncErrc
[SWS_TS_00902]	Definition of API class ara::tsync::TsyncException
[SWS_TS_00903]	Definition of API function ara::tsync::TsyncException::TsyncException
[SWS_TS_00904]	Definition of API class ara::tsync::TsyncErrorDomain
[SWS_TS_00905]	Definition of API function ara::tsync::TsyncErrorDomain::TsyncErrorDomain
[SWS_TS_00906]	Definition of API function ara::tsync::TsyncErrorDomain::Name
[SWS_TS_00907]	Definition of API function ara::tsync::TsyncErrorDomain::Message
[SWS_TS_00908]	Definition of API function ara::tsync::TsyncErrorDomain::ThrowAsException
[SWS_TS_00909]	Definition of API function ara::tsync::GetTsyncErrorDomain
[SWS_TS_00910]	Definition of API function ara::tsync::MakeErrorCode
[SWS_TS_01061]	Definition of API function ara::tsync::SynchronizedTimeBase Status::SynchronizedTimeBaseStatus
[SWS_TS_01062]	Definition of API function ara::tsync::SynchronizedTimeBase Status::~SynchronizedTimeBaseStatus
[SWS_TS_01204]	Definition of API function ara::tsync::OffsetTimeBaseProvider::OffsetTime BaseProvider
[SWS_TS_01300]	Definition of API function ara::tsync::ConsumerTimeBaseValidation Notification::~ConsumerTimeBaseValidationNotification
[SWS_TS_01301]	Definition of API function ara::tsync::ProviderTimeBaseValidation Notification::~ProviderTimeBaseValidationNotification
[SWS_TS_14175]	

Table D.1: Added Specification Items in R23-11

D.1.2 Changed Specification Items in R23-11

Number	Heading
[SWS_TS_00042]	
[SWS_TS_00047]	
[SWS_TS_00049]	
[SWS_TS_00052]	
[SWS_TS_00058]	
[SWS_TS_00064]	
[SWS_TS_00070]	
[SWS_TS_00071]	
[SWS_TS_00202]	
[SWS_TS_00420]	Definition of API function ara::tsync::ConsumerTimeBaseValidation Notification::SetSlaveTimingData
[SWS_TS_00421]	Definition of API function ara::tsync::ProviderTimeBaseValidation Notification::SetMasterTimingData
[SWS_TS_00422]	Definition of API function ara::tsync::ConsumerTimeBaseValidation Notification::SetPdelayInitiatorData
[SWS_TS_00423]	Definition of API function ara::tsync::ProviderTimeBaseValidation Notification::SetPdelayResponderData
[SWS_TS_00803]	
[SWS_TS_01001]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer
[SWS_TS_01002]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::~SynchronizedTimeBaseConsumer
[SWS_TS_01003]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer
[SWS_TS_01004]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::operator=
[SWS_TS_01005]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer
[SWS_TS_01006]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::operator=
[SWS_TS_01101]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider
[SWS_TS_01102]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider
[SWS_TS_01103]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::operator=
[SWS_TS_01104]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider
[SWS_TS_01105]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::operator=

△

Number	Heading
[SWS_TS_01106]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::~SynchronizedTimeBaseProvider
[SWS_TS_01107]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Set Time
[SWS_TS_01110]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Set RateCorrection
[SWS_TS_01201]	Definition of API function ara::tsync::OffsetTimeBaseProvider::OffsetTime BaseProvider
[SWS_TS_01202]	Definition of API function ara::tsync::OffsetTimeBaseProvider::OffsetTime BaseProvider
[SWS_TS_01203]	Definition of API function ara::tsync::OffsetTimeBaseProvider::operator=
[SWS_TS_01205]	Definition of API function ara::tsync::OffsetTimeBaseProvider::operator=
[SWS_TS_01206]	Definition of API function ara::tsync::OffsetTimeBaseProvider::~OffsetTime BaseProvider
[SWS_TS_01207]	Definition of API function ara::tsync::OffsetTimeBaseProvider::SetOffset Time
[SWS_TS_01209]	Definition of API function ara::tsync::OffsetTimeBaseProvider::SetRate Correction

Table D.2: Changed Specification Items in R23-11

D.1.3 Deleted Specification Items in R23-11

none

D.2 Change History of this document according to AUTOSAR Release R24-11

D.2.1 Added Specification Items in R24-11

Number	Heading
[SWS_TS_01266]	Definition of API type ara::tsync::TsyncErrorDomain::Errc
[SWS_TS_01267]	Definition of API type ara::tsync::TsyncErrorDomain::Exception
[SWS_TS_14176]	Definition of API function ara::tsync::SynchronizedTimeBaseStatus::Get CreationLocalTime
[SWS_TS_14178]	Initialization of the FVM
[SWS_TS_14179]	Usage of Secure Time Synchronization
[SWS_TS_14180]	Usage of Freshness Value
[SWS_TS_14181]	Check the configuration of Freshness Value

▽

△

Number	Heading
[SWS_TS_14182]	Algorithm of ICV generation
[SWS_TS_14183]	Secure Time Synchronization message sending
[SWS_TS_14184]	Secure Time Synchronization message build attempts
[SWS_TS_14185]	Secure Time Synchronization message receiving
[SWS_TS_14186]	Secure Time Synchronization verification attempts
[SWS_TS_14187]	Authentication Build Counter reached the threshold
[SWS_TS_14188]	Authentication Verify Counter reached the threshold
[SWS_TS_14189]	Freshness Value Manager is not available
[SWS_TS_14190]	RxDebounceCounter value is greater than 0
[SWS_TS_14191]	LogMessage Authentication Build Counter has reached the threshold value
[SWS_TS_14192]	LogMessage ICV verification attempt counter has reached the threshold value
[SWS_TS_14193]	LogMessage Verification of ICV failed
[SWS_TS_14194]	Definition of API class apext::tsync::FVContainer
[SWS_TS_14195]	Definition of API variable apext::tsync::FVContainer::length
[SWS_TS_14196]	Definition of API variable apext::tsync::FVContainer::value
[SWS_TS_14197]	Definition of API class apext::tsync::FVM
[SWS_TS_14198]	Definition of API function apext::tsync::FVM::GetRxFreshness
[SWS_TS_14199]	Definition of API function apext::tsync::FVM::GetTxFreshness
[SWS_TS_14200]	Definition of API function apext::tsync::FVM::Initialize
[SWS_TS_14201]	Definition of API function apext::tsync::FvmErrorDomain::~FvmErrorDomain
[SWS_TS_14202]	Definition of API enum apext::tsync::FvmErrc
[SWS_TS_14203]	Definition of API class apext::tsync::FvmException
[SWS_TS_14204]	Definition of API function apext::tsync::FvmException::FvmException
[SWS_TS_14205]	Definition of API class apext::tsync::FvmErrorDomain
[SWS_TS_14206]	Definition of API type apext::tsync::FvmErrorDomain::Errc
[SWS_TS_14207]	Definition of API type apext::tsync::FvmErrorDomain::Exception
[SWS_TS_14208]	Definition of API function apext::tsync::FvmErrorDomain::FvmErrorDomain
[SWS_TS_14209]	Definition of API function apext::tsync::FvmErrorDomain::Name
[SWS_TS_14210]	Definition of API function apext::tsync::FvmErrorDomain::Message
[SWS_TS_14211]	Definition of API function apext::tsync::FvmErrorDomain::ThrowAsException
[SWS_TS_14212]	Definition of API function apext::tsync::GetFvmErrorDomain
[SWS_TS_14213]	Definition of API function apext::tsync::MakeErrorCode
[SWS_TS_14214]	Security event context data definition: SEV_TSYN_ETH_ICV_GENERATION_FAILED
[SWS_TS_14215]	Security event context data definition: SEV_TSYN_ETH_ICV_VERIFICATION_FAILED
[SWS_TS_14216]	Security event context data definition: SEV_TSYN_ETH_FRESHNESS_NOT_AVAILABLE

▽

△

Number	Heading
[SWS_TS_14217]	Security event context data definition: SEV_TSYN_ETH_MSG_ SEQUENCE_ERROR
[SWS_TS_14218]	Get Time With Status
[SWS_TS_14219]	No user data available
[SWS_TS_14220]	Security events for Time Synchronization (TS)

Table D.3: Added Specification Items in R24-11

D.2.2 Changed Specification Items in R24-11

Number	Heading
[SWS_TS_00007]	
[SWS_TS_00041]	
[SWS_TS_00042]	
[SWS_TS_00043]	
[SWS_TS_00044]	
[SWS_TS_00045]	
[SWS_TS_00046]	
[SWS_TS_00048]	
[SWS_TS_00050]	
[SWS_TS_00051]	
[SWS_TS_00053]	
[SWS_TS_00054]	
[SWS_TS_00055]	
[SWS_TS_00056]	
[SWS_TS_00057]	
[SWS_TS_00058]	
[SWS_TS_00059]	
[SWS_TS_00060]	
[SWS_TS_00061]	
[SWS_TS_00062]	
[SWS_TS_00063]	
[SWS_TS_00070]	
[SWS_TS_00120]	
[SWS_TS_00127]	
[SWS_TS_00212]	
[SWS_TS_00213]	
[SWS_TS_00214]	

▽

△

Number	Heading
[SWS_TS_00215]	
[SWS_TS_00414]	Definition of API class ara::tsync::TimeMasterMeasurementType
[SWS_TS_00415]	Definition of API class ara::tsync::TimeSlaveMeasurementType
[SWS_TS_00416]	Definition of API class ara::tsync::PdelayInitiatorMeasurementType
[SWS_TS_00417]	Definition of API class ara::tsync::PdelayResponderMeasurementType
[SWS_TS_00419]	Definition of API class ara::tsync::ProviderTimeBaseValidationNotification
[SWS_TS_00420]	Definition of API function ara::tsync::ConsumerTimeBaseValidation Notification::SetSlaveTimingData
[SWS_TS_00421]	Definition of API function ara::tsync::ProviderTimeBaseValidation Notification::SetMasterTimingData
[SWS_TS_00422]	Definition of API function ara::tsync::ConsumerTimeBaseValidation Notification::SetPdelayInitiatorData
[SWS_TS_00423]	Definition of API function ara::tsync::ProviderTimeBaseValidation Notification::SetPdelayResponderData
[SWS_TS_00424]	
[SWS_TS_00425]	
[SWS_TS_00426]	
[SWS_TS_00427]	
[SWS_TS_00428]	Definition of API class ara::tsync::ConsumerTimeBaseValidationNotification
[SWS_TS_00803]	
[SWS_TS_00901]	Definition of API enum ara::tsync::TsyncErrc
[SWS_TS_00902]	Definition of API class ara::tsync::TsyncException
[SWS_TS_00903]	Definition of API function ara::tsync::TsyncException::TsyncException
[SWS_TS_00904]	Definition of API class ara::tsync::TsyncErrorDomain
[SWS_TS_00905]	Definition of API function ara::tsync::TsyncErrorDomain::TsyncErrorDomain
[SWS_TS_00906]	Definition of API function ara::tsync::TsyncErrorDomain::Name
[SWS_TS_00907]	Definition of API function ara::tsync::TsyncErrorDomain::Message
[SWS_TS_00908]	Definition of API function ara::tsync::TsyncErrorDomain::ThrowAsException
[SWS_TS_00909]	Definition of API function ara::tsync::GetTsyncErrorDomain
[SWS_TS_00910]	Definition of API function ara::tsync::MakeErrorCode
[SWS_TS_01001]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer
[SWS_TS_01002]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::~SynchronizedTimeBaseConsumer
[SWS_TS_01003]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer
[SWS_TS_01004]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::operator=
[SWS_TS_01005]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::SynchronizedTimeBaseConsumer

▽

△

Number	Heading
[SWS_TS_01006]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::operator=
[SWS_TS_01008]	Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::Get RateDeviation
[SWS_TS_01009]	Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::Get TimeWithStatus
[SWS_TS_01010]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::RegisterStatusChangeNotifier
[SWS_TS_01011]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterStatusChangeNotifier
[SWS_TS_01012]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::RegisterSynchronizationStateChangeNotifier
[SWS_TS_01013]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterSynchronizationStateChangeNotifier
[SWS_TS_01014]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::RegisterTimeLeapNotifier
[SWS_TS_01015]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterTimeLeapNotifier
[SWS_TS_01016]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::RegisterTimeValidationNotification
[SWS_TS_01017]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterTimeValidationNotification
[SWS_TS_01018]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::RegisterTimePrecisionMeasurementNotifier
[SWS_TS_01019]	Definition of API function ara::tsync::SynchronizedTimeBase Consumer::UnregisterTimePrecisionMeasurementNotifier
[SWS_TS_01053]	Definition of API function ara::tsync::SynchronizedTimeBaseStatus::Get SynchronizationStatus
[SWS_TS_01054]	Definition of API function ara::tsync::SynchronizedTimeBaseStatus::GetLeap Jump
[SWS_TS_01055]	Definition of API function ara::tsync::SynchronizedTimeBaseStatus::Get CreationTime
[SWS_TS_01056]	Definition of API function ara::tsync::SynchronizedTimeBaseStatus::GetUser Data
[SWS_TS_01057]	Definition of API function ara::tsync::SynchronizedTimeBase Status::SynchronizedTimeBaseStatus
[SWS_TS_01058]	Definition of API function ara::tsync::SynchronizedTimeBase Status::SynchronizedTimeBaseStatus
[SWS_TS_01059]	Definition of API function ara::tsync::SynchronizedTimeBase Status::operator=
[SWS_TS_01060]	Definition of API function ara::tsync::SynchronizedTimeBase Status::operator=
[SWS_TS_01061]	Definition of API function ara::tsync::SynchronizedTimeBase Status::SynchronizedTimeBaseStatus

▽

△

Number	Heading
[SWS_TS_01062]	Definition of API function ara::tsync::SynchronizedTimeBase Status::~SynchronizedTimeBaseStatus
[SWS_TS_01100]	Definition of API class ara::tsync::SynchronizedTimeBaseProvider
[SWS_TS_01101]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider
[SWS_TS_01102]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider
[SWS_TS_01103]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::operator=
[SWS_TS_01104]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::SynchronizedTimeBaseProvider
[SWS_TS_01105]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::operator=
[SWS_TS_01106]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::~SynchronizedTimeBaseProvider
[SWS_TS_01107]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Set Time
[SWS_TS_01108]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::UpdateTime
[SWS_TS_01109]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Get CurrentTime
[SWS_TS_01110]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Set RateCorrection
[SWS_TS_01111]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Get RateDeviation
[SWS_TS_01112]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Set UserData
[SWS_TS_01113]	Definition of API function ara::tsync::SynchronizedTimeBaseProvider::Get UserData
[SWS_TS_01114]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::RegisterTimeValidationNotification
[SWS_TS_01115]	Definition of API function ara::tsync::SynchronizedTimeBase Provider::UnregisterTimeValidationNotification
[SWS_TS_01251]	Definition of API type ara::tsync::Timestamp
[SWS_TS_01260]	Definition of API class ara::tsync::TimeBase
[SWS_TS_01261]	Definition of API type ara::tsync::TimeBase::rep
[SWS_TS_01262]	Definition of API type ara::tsync::TimeBase::period
[SWS_TS_01263]	Definition of API type ara::tsync::TimeBase::duration
[SWS_TS_01264]	Definition of API type ara::tsync::TimeBase::time_point
[SWS_TS_01265]	Definition of API variable ara::tsync::TimeBase::is_steady
[SWS_TS_01300]	Definition of API function ara::tsync::ConsumerTimeBaseValidation Notification::~ConsumerTimeBaseValidationNotification
[SWS_TS_01301]	Definition of API function ara::tsync::ProviderTimeBaseValidation Notification::~ProviderTimeBaseValidationNotification

▽

△

Number	Heading
[SWS_TS_01403]	Definition of API variable ara::tsync::TimePrecisionMeasurement::timeBase Status
[SWS_TS_14140]	Definition of API variable ara::tsync::TimeMasterMeasurementType::precise OriginTimestamp
[SWS_TS_14141]	Definition of API variable ara::tsync::TimeMasterMeasurementType::sync EgressTimestamp
[SWS_TS_14142]	Definition of API variable ara::tsync::TimeMasterMeasurement Type::sequenceId
[SWS_TS_14150]	Definition of API variable ara::tsync::TimeSlaveMeasurementType::precise OriginTimestamp
[SWS_TS_14151]	Definition of API variable ara::tsync::TimeSlaveMeasurementType::reference GlobalTimestamp
[SWS_TS_14152]	Definition of API variable ara::tsync::TimeSlaveMeasurementType::sync IngressTimestamp
[SWS_TS_14153]	Definition of API variable ara::tsync::TimeSlaveMeasurement Type::correctionField
[SWS_TS_14154]	Definition of API variable ara::tsync::TimeSlaveMeasurementType::reference LocalTimestamp
[SWS_TS_14155]	Definition of API variable ara::tsync::TimeSlaveMeasurementType::pDelay
[SWS_TS_14156]	Definition of API variable ara::tsync::TimeSlaveMeasurementType::sequence Id
[SWS_TS_14160]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::requestOriginTimestamp
[SWS_TS_14161]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::responseReceiptTimestamp
[SWS_TS_14162]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::requestReceiptTimestamp
[SWS_TS_14163]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::responseOriginTimestamp
[SWS_TS_14164]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::referenceLocalTimestamp
[SWS_TS_14165]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::referenceGlobalTimestamp
[SWS_TS_14166]	Definition of API variable ara::tsync::PdelayInitiatorMeasurementType::p Delay
[SWS_TS_14167]	Definition of API variable ara::tsync::PdelayInitiatorMeasurement Type::sequenceId
[SWS_TS_14170]	Definition of API variable ara::tsync::PdelayResponderMeasurement Type::requestReceiptTimestamp
[SWS_TS_14171]	Definition of API variable ara::tsync::PdelayResponderMeasurement Type::responseOriginTimestamp
[SWS_TS_14172]	Definition of API variable ara::tsync::PdelayResponderMeasurement Type::referenceLocalTimestamp

▽

△

Number	Heading
[SWS_TS_14173]	Definition of API variable ara::tsync::PdelayResponderMeasurement Type::referenceGlobalTimestamp
[SWS_TS_14174]	Definition of API variable ara::tsync::PdelayResponderMeasurement Type::sequenceId
[SWS_TS_14175]	

Table D.4: Changed Specification Items in R24-11

D.2.3 Deleted Specification Items in R24-11

Number	Heading
[SWS_TS_00047]	
[SWS_TS_00049]	
[SWS_TS_00052]	
[SWS_TS_00071]	
[SWS_TS_00129]	
[SWS_TS_00131]	
[SWS_TS_00801]	
[SWS_TS_01007]	Definition of API function ara::tsync::SynchronizedTimeBaseConsumer::GetCurrentTime
[SWS_TS_01200]	Definition of API class ara::tsync::OffsetTimeBaseProvider
[SWS_TS_01201]	Definition of API function ara::tsync::OffsetTimeBaseProvider::OffsetTimeBaseProvider
[SWS_TS_01202]	Definition of API function ara::tsync::OffsetTimeBaseProvider::OffsetTimeBaseProvider
[SWS_TS_01203]	Definition of API function ara::tsync::OffsetTimeBaseProvider::operator=
[SWS_TS_01204]	Definition of API function ara::tsync::OffsetTimeBaseProvider::OffsetTimeBaseProvider
[SWS_TS_01205]	Definition of API function ara::tsync::OffsetTimeBaseProvider::operator=
[SWS_TS_01206]	Definition of API function ara::tsync::OffsetTimeBaseProvider::~OffsetTimeBaseProvider
[SWS_TS_01207]	Definition of API function ara::tsync::OffsetTimeBaseProvider::SetOffsetTime
[SWS_TS_01208]	Definition of API function ara::tsync::OffsetTimeBaseProvider::GetCurrentTime
[SWS_TS_01209]	Definition of API function ara::tsync::OffsetTimeBaseProvider::SetRateCorrection
[SWS_TS_01210]	Definition of API function ara::tsync::OffsetTimeBaseProvider::GetRateDeviation
[SWS_TS_01211]	Definition of API function ara::tsync::OffsetTimeBaseProvider::SetUserData

▽

△

Number	Heading
[SWS_TS_01212]	Definition of API function ara::tsync::OffsetTimeBaseProvider::GetUserData
[SWS_TS_01213]	Definition of API function ara::tsync::OffsetTimeBaseProvider::RegisterTimeValidationNotification
[SWS_TS_01214]	Definition of API function ara::tsync::OffsetTimeBaseProvider::UnregisterTimeValidationNotification

Table D.5: Deleted Specification Items in R24-11

D.2.4 Added Constraints in R24-11

Number	Heading
[SWS_TS_CONSTR_00001]	Configurable Namespace for TimeSynchronization

Table D.6: Added Constraints in R24-11

D.2.5 Changed Constraints in R24-11

none

D.2.6 Deleted Constraints in R24-11

none