Document Title	Specification of Raw Data Stream
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1098

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R25-11

Document Change History

Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	• Renamed RawErrorDomain -> RdsErrorDomain • Editorial changes and bugfixes
2024-11-27	R24-11	AUTOSAR Release Management	• Added API for IEEE 1722 Audio Video Transport Protocol streams • Editorial changes and bugfixes
2023-11-23	R23-11	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the functional cluster Raw Data Stream for the AUTOSAR Adaptive Platform. Sometimes it could be necessary for the application software to be able to process raw binary data streams sent over a communication channel. In some cases a raw binary data stream is handled as a continuing sequence of bytes where the data is not typed (byte stream). So serialization of the data is not necessary. In some other cases one received Ethernet frame (datagram) is processed as an atomic unit of a stream, where parts of the header information need to be forwarded as typed data, and the according payload as raw binary data (datagram stream).

This functional cluster specifies an interface to support processing of raw binary data streams. Generally, the term Raw Data Stream is used to represent a raw binary data stream.

A Raw Data Streaming interface is statically defined and dependent on the underlying network protocol. The modeling for the Raw Data Streaming Interface supports the following network protocols:

- Raw Data Streaming over Ethernet using IP based protocols (network layer)
- Raw Data Streaming over Ethernet using IEEE1722 protocol (data link layer)

Raw Data Streaming over Ethernet using IP based protocols (network layer)

Raw Data Streams using IP based protocols use TCP/IP sockets as transport layer. Both unicast and multicast socket connections shall be supported. The TCP/IP sockets can use both TCP or UDP as transport protocol. TCP is the natural choice for Raw Data Streams using IP protocol, since it is a reliable stream oriented protocol. However, UDP shall also be supported when an unreliable connection is acceptable for the application.

The integration of the Raw Data Streaming Interface and Adaptive Applications is done in the deployment phase, by specifying various attributes and parameters for the TCP/IP socket connections that shall be used for the Raw Data Stream which is using IP based protocols.

Secure communication can be achieved by applying TLS or IPSec protocols in the middleware. Also access control imposed by the IAM can be applied for Raw Data Streams using IP based protocols.

Raw Data Streaming over Ethernet using IEEE1722 protocol (data link layer)

Raw Data Streams using IEEE1722 protocol use data link sockets (ISO OSI layer 2) as transport layer. Both MAC unicast and MAC multicast socket connections shall be supported. The data link sockets can use the following IEEE1722 subtypes (specified

by [1, IEEE1722]) as transport protocol: AAF, 61883_IIDC, RVF, CRF, TSCF and NTSCF

The integration of the Raw Data Streaming Interface and Adaptive Applications is done in the deployment phase, by specifying various attributes and parameters for the data link socket connections that shall be used for the Raw Data Stream which is using IEEE1722 protocol. A datagram is interchanged between the Raw Data Stream and an application, where the IEEE1722 header information is handled as typed data types and the payload as raw binary data bytes.

Secure communication can be achieved by using MACSec protocol in the middleware. Also access control imposed by the IAM can be applied for Raw Data Streams using IEEE1722 protocol.

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 [2].

Abbreviation / Acronym:	Description:
MAC address	Medium access control address (MAC addresses are used in the medium access control protocol sublayer of the data link layer (ISO OSI layer 2))
IP	Internet Protocol
SOME/IP	Scalable service-Oriented MiddlewarE over IP
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
E2E	End-to-end communication protection
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
61883_IIDC	IEC 61883/IIDC format as defined by [1, IEEE1722]
AAF	AVTP Audio Format as defined by [1, IEEE1722]
RVF	Raw Video Format as defined by [1, IEEE1722]
CRF	Control Reference Format as defined by [1, IEEE1722]
TSCF	Time Sensitive Control Format as defined by [1, IEEE1722]
NTSCF	None Time Sensitive Control Format as defined by [1, IEEE1722]

Table 2.1: Acronyms and Abbreviations

2.1 Definitions

2.1.1 MAC unicast socket connection

Definition: A "MAC unicast socket connection" is a communication via a socket (provided by the operation system) that is bound to a data link layer (ISO OSI layer 2), where the used address is a media access control address. Via this address exactly one counter communication partner can be reached (unicast).

2.1.2 MAC multicast socket connection

Definition: A "MAC multicast socket connection" is a communication via a socket (provided by the operation system) that is bound to a data link layer (ISO OSI layer 2), where the used address is a media access control address. Via this address one or more counter communication partner can be reached (multicast).

2.1.3 AVTPDU-common-header

Definition: An "AVTPDU-common-header" is defined by [1, IEEE1722] and represents the first 12 bits (subtype (8 bits), header specific (1 bit), version (3 bits) of a

AVTPDU-header. The AVTPDU-common-header contains the basic fields that all formats of AVTP stream data subtypes share.

2.1.4 AVTPDU-common-stream-header

Definition: An "AVTPDU-common-stream-header" is defined by [1, IEEE1722] and expands the AVTPDU-common-header used by a subset of AVTP stream data subtypes (e.g. 61883_IIDC, AAF, TSCF, RVF).

2.1.5 AVTPDU-common-control-header

Definition: An "AVTPDU-common-stream-header" is defined by [1, IEEE1722] and expands the AVTPDU-common-header used by a subset of AVTP stream data subtypes (e.g. ADP).

2.1.6 AVTPDU-alternative-header

Definition: An "AVTPDU-alternative-header" is defined by [1, IEEE1722] and used for AVTP stream data subtypes that do not exhibit the commonalities shared between formats that use the AVTPDU-common-stream-header or AVTPDU-common-control-headers. For example, CRF and NTSCF subtypes uses the AVTPDU-alternative-header.

2.1.7 AVTPDU-payload

Definition: An "AVTPDU-payload" is defined by [1, IEEE1722] and represents the second part of an AVTPDU. The AVTPDU-payload carry data of subtype encoded in the AVTPDU-header. For example, an AAF-payload carry audio data (i.e. audio samples).

2.1.8 Stream data producer

Definition: A "stream data producer" represent an end node in an Ethernet network which produces (continously) data. The data is transmitted via a stream and received by 1 or multiple end nodes (stream data consumer).
Note: The term "talker" is synonymous with "stream data producer".

2.1.9 Stream data consumer

Definition: A "stream data consumer" represent an end node in an Ethernet network which consumes (continously) data. The data is received via a stream.
Note: The term "listener" is synonymous with "stream data consumer".

2.1.10 AVTP presentation time

Definition: The "AVTP presentation time" is defined by [1, IEEE1722] and represents the gPTP time at which designated data within an AVTPDU payload is transferred to a time-sensitive application of an stream data consumer. An AVTPDU-header of header format AVTPDU-common-stream-header carries the presentation time as "avtp_timestamp" according to [1, IEEE1722]. AVTP presentation time is calculated as "TavtpPresentationTime" = "TcurrentGlobalTime" + "TmaxTransitTime". Please note: presentation time does not cover format conversion time and processing time of the receiving time-sensitive application (see [1, IEEE1722] figure 6 "Figure 6 - AVTP Timing Reference Planes").

2.1.11 Max transit time

Definition: "Max transit time" is defined by [1, IEEE1722]. The basic method to calculate an appropriate "Max Transit Time" is to take the worst-case transit time from the stream data producer (talker) to a stream data consumer (listener) and choose a max transit time that is greater than or equal to the largest worst-case transit time.

2.1.12 Media clock

Definition: "Media clock" is defined by [1, IEEE1722] and represents an entity which generate a rate (e.g. precise hardware clock with an constant rate (e.g. 48kHz). The media clock is hosted by the media clock provider.

2.1.13 Media clock provider

Definition: A "media clock provider" is an end node in the network which hosts an media clock. The media clock provider transmit an IEEE1722 stream to 1 or multiple media clock consumer. The IEEE1722 stream is of subtype CRF (clock reference format) and contain several presentation timestamps which correlates to the media clock rate.

2.1.14 Media clock consumer

Definition: A "media clock consumer" is an end node in the network which receive a IEEE1722 stream of subtype CRF (clock reference format) from a media clock provider. The media clock consumer perform a recovery of its media clock (e.g. PLL) based on the received encapsulated data from the media clock provider.

2.1.15 ACF-stream

Definition: An "ACF-stream" represents an IEEE1722 stream of subtype TSCF (time-synchronous control format) or NTSCF(non-time-synchronous control format) which transport encapsulated bus frames as ACF-messages (e.g. `ACF_CAN`) within its AVTPDU-payload (ACF-payload).

2.1.16 ACF-message

Definition: An "ACF-message" is defined by [1, IEEE1722]) and represents an encapsulated bus frame of a certain kind of bus type (e.g. CAN). The bus frame is encapsulate with an corresponding ACF-message type (e.g. `ACF_CAN`). The ACF-message consist of an ACF-message-header and an ACF-message-payload. Multipe ACF-messages of different ACF-message types could form an ACF-payload which is transported within the same ACF-stream.

2.1.17 ACF-message-header

Definition: An "ACF-message-header" represents the first part of an ACF-message. The first 7 bits represents the ACF-message type. Several ACF-message types are specified by [1, IEEE1722] (e.g. `ACF_CAN`, `ACF_LIN`). The following 9 bits represents the lenght of the subsequential ACF-message-payload.

2.1.18 ACF-message-payload

Definition: An "ACF-message-payload" is defined by [1, IEEE1722] and represents the second part of an ACF-message. The ACF-message-payload carry data of ACF-message type encoded in the ACF-message-header. For example, an ACF_CAN-payload carry an CAN2.0 frame.

3 Related documentation

3.1 Input documents & related standards and norms

[1] IEEE Standard 1722-2016 - IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core

[4] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[5] Requirements on Communication Management
AUTOSAR_AP_RS_CommunicationManagement

[6] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[7] Requirements on IEEE1722
AUTOSAR_FO_RS_IEEE1722

[8] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[9] Specification of Communication Management
AUTOSAR_AP_SWS_CommunicationManagement

[10] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

3.2 Further applicable specification

AUTOSAR provides a core specification [3] which is also applicable for this functional cluster. The chapter [3] 7.1 *"General requirements for all Functional Clusters"* shall be considered an additional and required specification for implementing this functional cluster.

AUTOSAR provides a general specification [4, RS General] which is also applicable for the `Raw Data Stream` functional cluster. The specification SWS General shall be considered as additional and required specification for implementation of the `Raw Data Stream` functional cluster.

Currently, the specific requirements for the `Raw Data Stream` functional cluster are part of [5, RS Communication Management].

4 Constraints and assumptions

4.1 Known limitations

The current solution does not support any runtime variance in terms of network topology, such as service discovery functionality, which means that the RawDataStreams has to be configured statically on the same ECU as the application. Dynamic configuration and runtime functionality will be added in future releases if needed.

Raw Data Streams do not provide handling of safety protection. Safety communication mechanisms like E2E has to be applied on application level.

Raw Data Streams over Ethernet using IP based protocols (network layer)

The multicast support for Raw Data Streams using IP based protocols is limited to one-to-many, i.e. a server can send data to multiple clients using multicast IP address, but only receive data from one client, using the unicast IP address. Also multicast shall only be used with UDP. For TCP connections, only 1-to-1 connections are supported, i.e. multiple clients to one server is not supported.

Raw Data Streams over Ethernet using IEEE1722 protocol (data link layer)

A Raw Data Streams using IEEE1722 protocol support the following IEEE1722 subtypes (specified by [1, IEEE1722]) as transport protocol: AAF, 61883_IIDC, RVF, CRF, TSCF and NTSCF. Support of other subtypes specified by [1, IEEE1722] may be added in future.

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 [6].

5.1 Provided Interfaces

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

Interface	*Functional Cluster*	*Purpose*
No provided interfaces		

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.

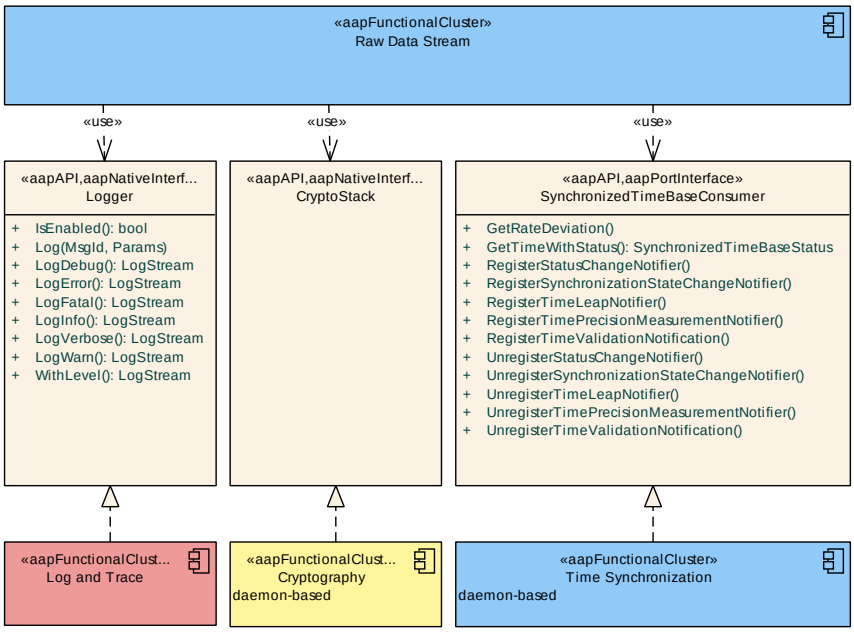

Figure 5.1: Interfaces required by Raw Data Stream from other Functional Clusters

Figure 5.1 shows interfaces required by `Raw Data Stream` functional cluster from other Functional Clusters within the AUTOSAR Adaptive Platform. Table 5.2 provides a complete list of required interfaces from other Functional Clusters within the AUTOSAR Adaptive Platform.

Functional Cluster	***Interface***	***Purpose***
Cryptography	CryptoStack	This interface may be used to establish encrypted connections.
Log and Trace	Logger	`Raw Data Stream` uses this interface to log standardized messages.
Time Synchronization	SynchronizedTimeBaseConsumer	This interface is used to determine the current synchronized global time for IEEE1722-based communication.

Table 5.2: Interfaces required from other Functional Clusters

6 Requirements Tracing

The following tables reference the requirements specified in the Requirements on Communication Management document [5], the General Requirements specific to Adaptive Platform [4], and the Requirements on IEEE1722 [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
[FO_RS_IEEE1722_00002]	IEEE1722Tp module handling of IEEE1722 streams	[SWS_RDS_10542] [SWS_RDS_10543] [SWS_RDS_10544]
[RS_AP_00119]	Return values / application errors	[SWS_RDS_90315] [SWS_RDS_90317] [SWS_RDS_90326] [SWS_RDS_90328]
[RS_AP_00120]	Method and Function names	[SWS_RDS_11292] [SWS_RDS_11294] [SWS_RDS_11295] [SWS_RDS_11296] [SWS_RDS_11297] [SWS_RDS_11298] [SWS_RDS_11299] [SWS_RDS_11326] [SWS_RDS_11327] [SWS_RDS_90313] [SWS_RDS_90314] [SWS_RDS_90315] [SWS_RDS_90316] [SWS_RDS_90317] [SWS_RDS_90324] [SWS_RDS_90325] [SWS_RDS_90326] [SWS_RDS_90327] [SWS_RDS_90328]
[RS_AP_00121]	Parameter names	[SWS_RDS_11292] [SWS_RDS_11296] [SWS_RDS_11297] [SWS_RDS_11299] [SWS_RDS_90314] [SWS_RDS_90315] [SWS_RDS_90325] [SWS_RDS_90326]
[RS_AP_00122]	Type names	[SWS_RDS_11291] [SWS_RDS_11293] [SWS_RDS_12367]
[RS_AP_00127]	Usage of ara::core types	[SWS_RDS_11291] [SWS_RDS_11293] [SWS_RDS_12367]
[RS_AP_00129]	Public types defined by functional clusters shall be designed to allow implementation without dynamic memory allocation	[SWS_RDS_90313] [SWS_RDS_90314] [SWS_RDS_90324] [SWS_RDS_90325]
[RS_AP_00130]	AUTOSAR Adaptive Platform shall represent a rich and modern programming environment	[SWS_RDS_11291] [SWS_RDS_11292] [SWS_RDS_11293] [SWS_RDS_11294] [SWS_RDS_11295] [SWS_RDS_11296] [SWS_RDS_11297] [SWS_RDS_11298] [SWS_RDS_11299] [SWS_RDS_11326] [SWS_RDS_11327] [SWS_RDS_12367]
[RS_AP_00145]	Availability of special member functions	[SWS_RDS_10482] [SWS_RDS_10483] [SWS_RDS_10512] [SWS_RDS_11303] [SWS_RDS_11304] [SWS_RDS_11305] [SWS_RDS_11306] [SWS_RDS_11312] [SWS_RDS_11313] [SWS_RDS_11314] [SWS_RDS_11315] [SWS_RDS_11316] [SWS_RDS_11317] [SWS_RDS_90314] [SWS_RDS_90315] [SWS_RDS_90316] [SWS_RDS_90317] [SWS_RDS_90325] [SWS_RDS_90326] [SWS_RDS_90327] [SWS_RDS_90328]
[RS_AP_00146]	Classes whose construction requires interaction by the ARA framework	[SWS_RDS_11294] [SWS_RDS_90313] [SWS_RDS_90324]
[RS_AP_00147]	Classes that are created with an InstanceSpecifier as an argument are not copyable, but at most movable.	[SWS_RDS_11303] [SWS_RDS_11304] [SWS_RDS_11305] [SWS_RDS_11306] [SWS_RDS_11316] [SWS_RDS_11317]

▽

△

Requirement	Description	Satisfied by
[RS_AP_00158]	IAM access violations	[SWS_RDS_10540] [SWS_RDS_90007]
[RS_AP_00170]	InstanceSpecifierMappingIntegrity Violation	[SWS_RDS_10482] [SWS_RDS_11312]
[RS_AP_00171]	PortInterfaceMappingViolation	[SWS_RDS_10482] [SWS_RDS_11312]
[RS_AP_00172]	ProcessMappingViolation	[SWS_RDS_10482] [SWS_RDS_11312]
[RS_AP_00173]	InstanceSpecifierAlreadyInUse Violation	[SWS_RDS_10482] [SWS_RDS_11312]
[RS_CM_00410]	The Communication Management shall provide an API to support reading and writing raw data streams that has no datatype information	[SWS_RDS_10476] [SWS_RDS_10477] [SWS_RDS_10478] [SWS_RDS_10479] [SWS_RDS_10480] [SWS_RDS_10481] [SWS_RDS_10482] [SWS_RDS_10483] [SWS_RDS_10484] [SWS_RDS_10485] [SWS_RDS_10486] [SWS_RDS_10487] [SWS_RDS_10508] [SWS_RDS_10511] [SWS_RDS_10512] [SWS_RDS_10513] [SWS_RDS_10514] [SWS_RDS_10515] [SWS_RDS_10516] [SWS_RDS_10517] [SWS_RDS_10518] [SWS_RDS_10519] [SWS_RDS_10520] [SWS_RDS_11300] [SWS_RDS_11301] [SWS_RDS_11302] [SWS_RDS_11303] [SWS_RDS_11304] [SWS_RDS_11305] [SWS_RDS_11306] [SWS_RDS_11307] [SWS_RDS_11309] [SWS_RDS_11310] [SWS_RDS_11311] [SWS_RDS_11312] [SWS_RDS_11313] [SWS_RDS_11314] [SWS_RDS_11315] [SWS_RDS_11316] [SWS_RDS_11317] [SWS_RDS_11318] [SWS_RDS_11319] [SWS_RDS_11320] [SWS_RDS_11322] [SWS_RDS_11323] [SWS_RDS_11324] [SWS_RDS_11325] [SWS_RDS_90216] [SWS_RDS_90217] [SWS_RDS_90311] [SWS_RDS_90321] [SWS_RDS_99004] [SWS_RDS_99006]

▽

△

Requirement	Description	Satisfied by
[RS_CM_00411]	Application developers shall be able to send and receive raw binary data streams independent of the underlying network protocol	[SWS_RDS_10476] [SWS_RDS_10477] [SWS_RDS_10478] [SWS_RDS_10479] [SWS_RDS_10480] [SWS_RDS_10481] [SWS_RDS_10482] [SWS_RDS_10483] [SWS_RDS_10484] [SWS_RDS_10485] [SWS_RDS_10486] [SWS_RDS_10487] [SWS_RDS_10508] [SWS_RDS_10511] [SWS_RDS_10512] [SWS_RDS_10513] [SWS_RDS_10514] [SWS_RDS_10515] [SWS_RDS_10516] [SWS_RDS_10517] [SWS_RDS_10518] [SWS_RDS_10519] [SWS_RDS_10520] [SWS_RDS_11300] [SWS_RDS_11301] [SWS_RDS_11302] [SWS_RDS_11303] [SWS_RDS_11304] [SWS_RDS_11305] [SWS_RDS_11306] [SWS_RDS_11307] [SWS_RDS_11309] [SWS_RDS_11310] [SWS_RDS_11311] [SWS_RDS_11312] [SWS_RDS_11313] [SWS_RDS_11314] [SWS_RDS_11315] [SWS_RDS_11316] [SWS_RDS_11317] [SWS_RDS_11318] [SWS_RDS_11319] [SWS_RDS_11320] [SWS_RDS_11322] [SWS_RDS_11323] [SWS_RDS_11324] [SWS_RDS_11325] [SWS_RDS_90216] [SWS_RDS_90217] [SWS_RDS_90311] [SWS_RDS_90321] [SWS_RDS_99004] [SWS_RDS_99005] [SWS_RDS_99006]
[RS_CM_00412]	The Communication Management shall provide TCP/IP Sockets as network protocol for Raw Data Streams	[SWS_RDS_10476] [SWS_RDS_10477] [SWS_RDS_10478] [SWS_RDS_10479] [SWS_RDS_10480] [SWS_RDS_10482] [SWS_RDS_10483] [SWS_RDS_10484] [SWS_RDS_10485] [SWS_RDS_10486] [SWS_RDS_10487] [SWS_RDS_10508] [SWS_RDS_10511] [SWS_RDS_10512] [SWS_RDS_10513] [SWS_RDS_10514] [SWS_RDS_10515] [SWS_RDS_10516] [SWS_RDS_10517] [SWS_RDS_10518] [SWS_RDS_10519] [SWS_RDS_10520] [SWS_RDS_11300] [SWS_RDS_11301] [SWS_RDS_11302] [SWS_RDS_11303] [SWS_RDS_11304] [SWS_RDS_11305] [SWS_RDS_11306] [SWS_RDS_11307] [SWS_RDS_11309] [SWS_RDS_11310] [SWS_RDS_11312] [SWS_RDS_11313] [SWS_RDS_11314] [SWS_RDS_11315] [SWS_RDS_11316] [SWS_RDS_11317] [SWS_RDS_11318] [SWS_RDS_11319] [SWS_RDS_11320] [SWS_RDS_11322] [SWS_RDS_11323] [SWS_RDS_11324] [SWS_RDS_11325] [SWS_RDS_90216] [SWS_RDS_99004] [SWS_RDS_99005]

▽

△

Requirement	Description	Satisfied by
[RS_CM_00413]	Support of IEEE1722 Stream handling	[SWS_RDS_10525] [SWS_RDS_10526] [SWS_RDS_10527] [SWS_RDS_10528] [SWS_RDS_10529] [SWS_RDS_10530] [SWS_RDS_10531] [SWS_RDS_10532] [SWS_RDS_10533] [SWS_RDS_10534] [SWS_RDS_10535] [SWS_RDS_10536] [SWS_RDS_10537] [SWS_RDS_10538] [SWS_RDS_10539] [SWS_RDS_10541] [SWS_RDS_10545] [SWS_RDS_10546] [SWS_RDS_10547] [SWS_RDS_10548] [SWS_RDS_10549] [SWS_RDS_10550] [SWS_RDS_10551] [SWS_RDS_10552] [SWS_RDS_10553] [SWS_RDS_10554] [SWS_RDS_10555] [SWS_RDS_10556] [SWS_RDS_10557] [SWS_RDS_10558] [SWS_RDS_10559] [SWS_RDS_10560] [SWS_RDS_10561] [SWS_RDS_10562] [SWS_RDS_10563] [SWS_RDS_10564] [SWS_RDS_10565] [SWS_RDS_10566] [SWS_RDS_10567] [SWS_RDS_10568] [SWS_RDS_10569] [SWS_RDS_10570] [SWS_RDS_10571] [SWS_RDS_90225] [SWS_RDS_90226] [SWS_RDS_90227] [SWS_RDS_90228] [SWS_RDS_90229] [SWS_RDS_90230] [SWS_RDS_90231] [SWS_RDS_90232] [SWS_RDS_90233] [SWS_RDS_90234] [SWS_RDS_90235] [SWS_RDS_90236] [SWS_RDS_90237] [SWS_RDS_90238] [SWS_RDS_90239] [SWS_RDS_90240] [SWS_RDS_90241] [SWS_RDS_90242] [SWS_RDS_90243] [SWS_RDS_90244] [SWS_RDS_90245] [SWS_RDS_90246] [SWS_RDS_90247] [SWS_RDS_90248] [SWS_RDS_90249] [SWS_RDS_90250] [SWS_RDS_90251] [SWS_RDS_90252] [SWS_RDS_90253] [SWS_RDS_90254] [SWS_RDS_90255] [SWS_RDS_90256] [SWS_RDS_90257] [SWS_RDS_90258] [SWS_RDS_90259] [SWS_RDS_90260] [SWS_RDS_90261] [SWS_RDS_90262] [SWS_RDS_90263] [SWS_RDS_90264] [SWS_RDS_90265] [SWS_RDS_90266] [SWS_RDS_90267] [SWS_RDS_90268] [SWS_RDS_90269] [SWS_RDS_90270] [SWS_RDS_90271] [SWS_RDS_90272] [SWS_RDS_90273] [SWS_RDS_90274] [SWS_RDS_90275] [SWS_RDS_90276] [SWS_RDS_90277] [SWS_RDS_90278] [SWS_RDS_90279] [SWS_RDS_90280] [SWS_RDS_90281] [SWS_RDS_90282] [SWS_RDS_90283] [SWS_RDS_90284] [SWS_RDS_90285] [SWS_RDS_90286] [SWS_RDS_90287] [SWS_RDS_90288] [SWS_RDS_90289] [SWS_RDS_90290] [SWS_RDS_90291] [SWS_RDS_90292] [SWS_RDS_90293] [SWS_RDS_90294] [SWS_RDS_90295] [SWS_RDS_90296] [SWS_RDS_90297] [SWS_RDS_90298] [SWS_RDS_90299] [SWS_RDS_90300] [SWS_RDS_90301] [SWS_RDS_90302] [SWS_RDS_90303] [SWS_RDS_90304] [SWS_RDS_90305] [SWS_RDS_90306] [SWS_RDS_90307] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_RDS_90308] [SWS_RDS_90309] [SWS_RDS_90310] [SWS_RDS_90311] [SWS_RDS_90312] [SWS_RDS_90313] [SWS_RDS_90314] [SWS_RDS_90315] [SWS_RDS_90316] [SWS_RDS_90317] [SWS_RDS_90318] [SWS_RDS_90319] [SWS_RDS_90320] [SWS_RDS_90321] [SWS_RDS_90322] [SWS_RDS_90323] [SWS_RDS_90324] [SWS_RDS_90325] [SWS_RDS_90326] [SWS_RDS_90327] [SWS_RDS_90328] [SWS_RDS_90329] [SWS_RDS_90330] [SWS_RDS_90331] [SWS_RDS_90332]
[RS_CM_00801]	Secure communication shall be transmitted using secure channels	[SWS_RDS_90211] [SWS_RDS_90212] [SWS_RDS_90213] [SWS_RDS_90214] [SWS_RDS_90215]
[RS_CM_00803]	The assignment of communication to specific secure channels shall be configurable	[SWS_RDS_90212]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Raw Data Streaming using IP based protocols (network layer)

7.1.1 Raw Data Streaming Interface

The operations of the interface are synchronous. The default behavior is blocking, but a timeout handling shall be implemented to return the call with an error if the operation takes too long. The timeout values are applied as parameters to each operation. See the description for each operation below on how the timeout handling is applied.

The configuration of the Raw Data Streams is done by specifying credentials and parameters for the socket connections that shall be used for the Raw Data Stream, using `RawDataStreamMapping` and `EthernetRawDataStreamMapping`. The model elements and the parameters are described in *TPS_ManifestSpecification* [8].

Secure communication can be achieved by applying TLS or IPSec protocols in the middleware. Also access control imposed by the IAM can be applied for Raw Data Streams. All security functions are configurable in the deployment and mapping model of Raw Data Streaming Interface, see *TPS_ManifestSpecification* [8].

All security functions (TLS, IPSec, IAM) are configurable in the deployment and mapping model of Raw Data Streaming Interface, see *TPS_ManifestSpecification* [8].

An application can use the Raw Data Streaming API both as a client (connecting to a listening Raw Data Streaming service) or server (waiting for incoming connections from clients).

Figure 7.1 shows the logical view of the usage of RawDataStream instances.

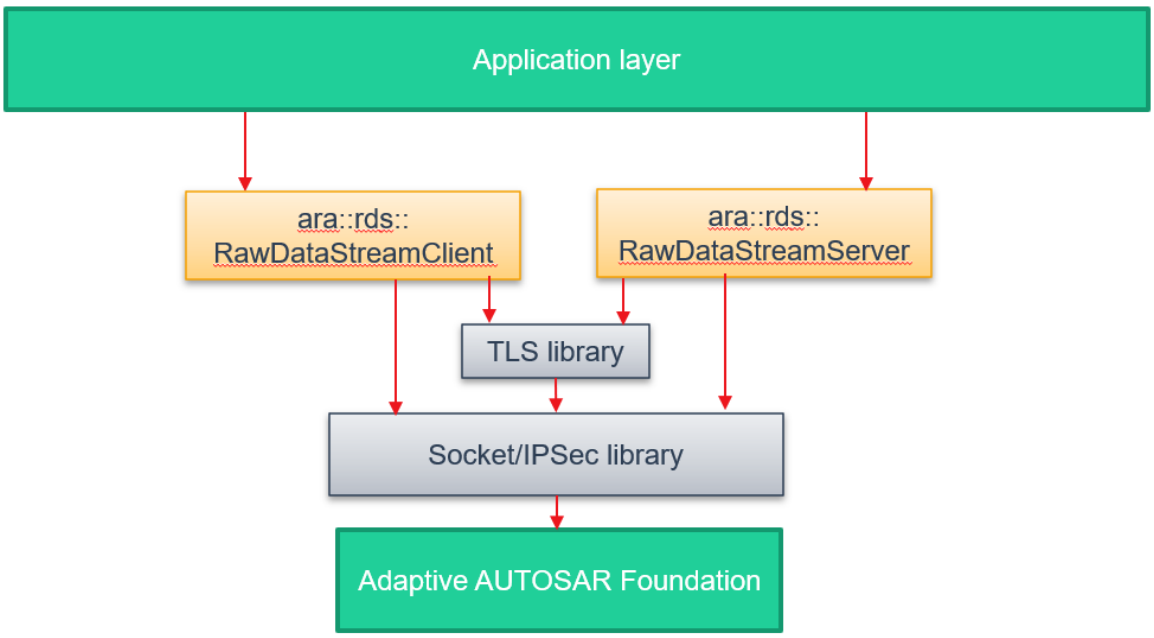

Figure 7.1: Raw Data Stream Logical View.

7.1.1.1 Use cases

The RawDataStream interface can be used in the following set-ups:

- Client (connect to) to an external non-AUTOSAR sensor providing raw data on a socket connection.
- Server (wait for a connection from) for an external non-AUTOSAR sensor providing raw data on a socket connection.
- Client or Server for another AUTOSAR external RawDataStream instance.

RawDataStream socket connections can be setup for UDP or TCP, Unicast or Multicast. Currently the use cases in fig 7.2 are supported.

See chapter 10 for information on the ethernet socket configuration parameters for the different use cases.

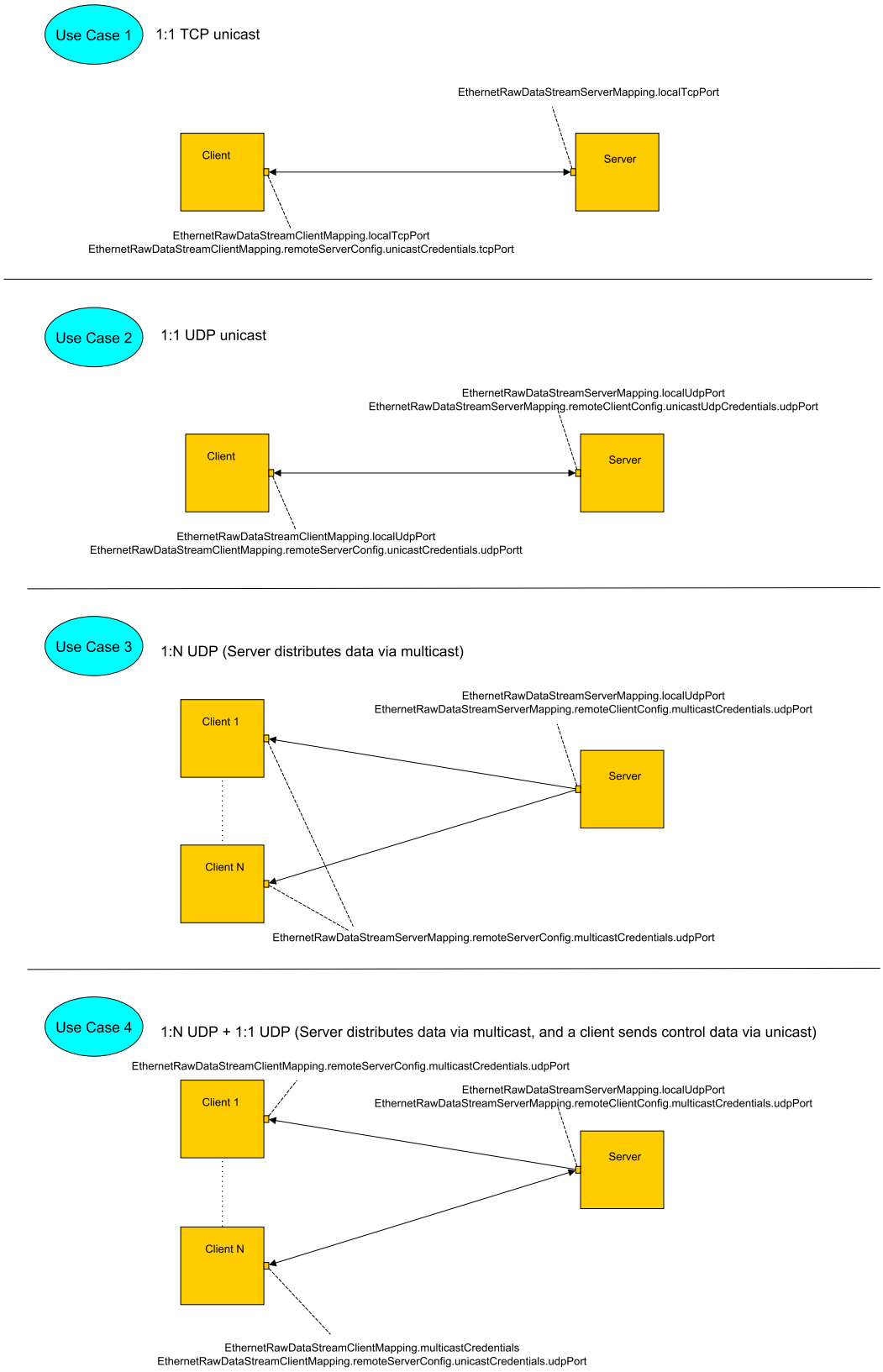

Figure 7.2: The currently supported use cases for Raw Data Streams, and which artifacts in the Deployment model that shall be used to configure the different use cases

7.1.2 Raw Data Streaming

For the IP based Raw Data Stream C++ API reference, see chapter 8.2 “Header: ara/rds/raw_data_stream.h”.

[SWS_RDS_10476] Defining a RawDataStream

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈To open a `RawDataStream` connection a `RawDataStream` instance is created. The constructor creates the necessary socket data structures for `RawDataStream` Communication, using the artifacts specified in the mapped `EthernetRawDataStreamClientMapping` and `EthernetRawDataStreamServerMapping`.⌋

The functionality of a `RawDataStream` for Client communication is realized in these four operations: Connect, Shutdown, ReadData and WriteData. A `RawDataStream` for Server Communication is realized in these four operations: WaitForConnection, Shutdown, ReadData and WriteData.

[SWS_RDS_10477] Connect stream link

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈Each invocation of the Connect operation for a TCP socket connection shall establish a communication link with a remote server that is listening for socket connections, The socket created in the `RawDataStream` instance shall be used for the connection. For UDP socket connections Connect shall do nothing.⌋

[SWS_RDS_99005] Wait for incoming connections

Upstream requirements: RS_CM_00411, RS_CM_00412

⌈Each invocation of the WaitForConnection operation shall wait for and accept incoming requests for establishment of a TCP communication link with a connecting remote client. The socket created and prepared in the RawDataStream instance shall be used for the connection. For UDP socket connections WaitForConnection shall do nothing.⌋

[SWS_RDS_10478] Shutdown stream link

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈Each invocation of the Shutdown operation shall destroy the communication link for the stream.⌋

[SWS_RDS_10508] Destructor behavior when Shutdown stream link

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈If the destructor is executed on an instance on which no Shutdown operation has been performed, the destructor shall perform Shutdown internally before the object is destroyed.⌋

[SWS_RDS_10479] Read data from stream

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈Each invocation of the ReadData operation shall request to read a number of bytes from the stream. The read data shall be moved to a buffer returned as result from the function, together with the actual number of bytes transferred.⌋

[SWS_RDS_10480] Write data to stream

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈Each invocation of the WriteData operation shall request to write a number of bytes to the stream and send it out on the socket connection. The actual number of bytes transferred shall be returned. It shall be possible to apply a timeout value for the operation. The operation shall write the data to the socket or internal buffer, and then return with the number of bytes written. For efficiency, the Write operation does not wait until data is actually sent on the bus, but the TCP data flow handling shall make sure that data is transmitted and received in the correct order. For UDP connections the order cannot be guaranteed.⌋

[SWS_RDS_99006] Timeout handling

Upstream requirements: RS_CM_00410, RS_CM_00411

⌈For all Connect, WaitForConnection, Read and Write `RawDataStream` operations a timeout value can be specified via a parameter in runtime. If no timeout parameter is given the operation shall block. If a timeout value is specified, and the operation does not finish within the specified time, an error code `RdsErrc::kCommunicationTimeout` shall be returned and the technical state of the `RawDataStream` connection shall be restored to the same as before the call was made.⌋

7.1.3 Security

7.1.3.1 Access Control via IAM

[SWS_RDS_90007] Restrictions on using RawDataStreams

Upstream requirements: RS_AP_00158

⌈If a `Process` calls

- the `RawDataStreamClient::Create()` constructor (see [SWS_RDS_10482])

or

- the `RawDataStreamServer::Create()` constructor (see [SWS_RDS_11312])

providing an InstanceSpecifier that does not identify a RPortPrototype referenced by a RawDataStreamMapping in the role portPrototype, that is mapped to the requesting Process in the role process, then this shall be treated as a violation.

⌋

7.1.3.1.1 Secure Communication

Raw Data Strem communication can be transported via TCP and UDP. Therefore different security mechanisms is available to secure the communication. The following security protocols are currently supported:

- TLS 1.2 (see [RFC5246])
- DTLS 1.2 (see [RFC6347])
- IPSec

7.1.3.1.2 Creation and use of secure channels for Raw Data Streaming

[SWS_RDS_90211] Secure UDP and TCP channel creation for TLS and DTLS

Upstream requirements: RS_CM_00801

⌈The Raw Data Stream software shall create secure UDP and TCP channels according to the input for all TlsSecureComProps as part of the EthernetRawDataStreamMapping.⌋

[SWS_RDS_90212] Using secure TLS, DTLS channels

Upstream requirements: RS_CM_00801, RS_CM_00803

⌈All communication triggered by a RawDataStream shall be sent via the respective secure channel according to the input. The appropriate secure channel is defined in the TlsSecureComProps as part of the EthernetRawDataStreamMapping that is mapped to an EthernetCommunicationConnector.⌋

7.1.3.1.3 (D)TLS for Raw Data Streaming

A (D)TLS secure channel may provide authenticity, integrity and confidentiality which may be used with raw data streaming.

The TLS and DTLS implementation should support the following cipher suites:

- TLS_PSK_WITH_NULL_SHA256 for authentic communication (see [RFC5487]).
- TLS_PSK_WITH_AES_128_GCM_SHA256 for confidential communication (see [RFC5487]).

[SWS_RDS_90213] TLS secure channel for raw data streams using reliable transport

Upstream requirements: RS_CM_00801

⌈A TLS secure channel shall be created and used if

- a `TlsSecureComProps` instance is part of a `EthernetRawDataStreamMapping` and is configured for transmission over “tcp” by assigning a `localTcpPort` in the `EthernetRawDataStreamMapping`.

⌋

[SWS_RDS_90214] DTLS secure channel for methods using unreliable transport

Upstream requirements: RS_CM_00801

⌈A DTLS secure channel shall be created and used if:

- a `TlsSecureComProps` instance is part of a `EthernetRawDataStreamMapping` and is configured for transmission over “udp” by assigning a `localUdpPort` in the `EthernetRawDataStreamMapping`.

⌋

[SWS_RDS_90215] IPsec secure channel between communication nodes and Transport of Raw Data Stream communication over an IPsec security association

Upstream requirements: RS_CM_00801

⌈An IPsec secure channel shall be created and used according to the requirements and constraints specified in [SWS_CM_90117] and [SWS_CM_90118], (see [9, SWS Communication Management]), by applying the `EthernetRawDataStreamMapping` to map to the `EthernetCommunicationConnector` that in turn references a `NetworkEndpoint` that contains an `IPSecConfig`.⌋

7.1.4 Raw Data Stream Interfaces

See chapter 7.1.1.1 for the different use cases that are supported.

7.1.4.1 Creation of a RawDataStream

[SWS_RDS_10511] Creation of RawDataStreamClient instance

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈A `RawDataStream` instance on the client side (`RawDataStreamClient` object) is created by calling the named exception-less RawDataStreamClient::Create() constructor takes an instance Specifier qualifying the wanted network binding and parameters for the instance.

If Remote Unicast Credentials (TCP or UDP) are defined for the client, the constructor shall create an endpoint for the communication, and store the handle in the created `RawDataStreamClient` object, to be used in the Read- and Write-operations for the `RawDataStreamClient` (for 1:1 use cases).

If Multicast Credentials (UDP) are defined for the client, the constructor shall create an endpoint for the communication, bind and join the multicast address and port specified in the MulticastCredentials.

For 1:N use cases this endpoint shall be used when RawDataStreamsClient.ReadData() is called, otherwise (for 1:1 use cases), the unicast endpoint shall be used for reading data.⌋

[SWS_RDS_10512] Creation of RawDataStreamServer instance

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145

⌈A `RawDataStream` instance on the server side (`RawDataStreamServer` object) is created by calling the named exception-less RawDataStreamServer::Create() constructor that takes an instance Specifier qualifying the wanted network credentials (UDP or TCP) for the instance.

A socket shall be created and bound to the address and port specified in the local credentials. In case of TCP it shall also mark the socket as passive and listen for connections (for use case 1:1 TCP unicast).

If Remote Unicast Credentials (UDP) are defined for the server, the constructor shall create an endpoint for the communication, and store the handle in the created `RawDataStreamServer` object, to be used in the Read and Write- operations for the `RawDataStreamServer` (for use case 1:1 UDP unicast).

If Multicast Credentials (UDP) are defined for the server, the constructor shall create an endpoint for the remote communication, bind and join the multicast address and port specified in the MulticastCredentials. In this case, this endpoint shall be used when RawDataStreams Server.WriteData() is called (for 1:N use cases), otherwise the unicast endpoint shall be used for writing data (for 1:1 use cases).⌋

7.1.4.2 Set up a RawDataStream connection

[SWS_RDS_10513] Setup RawDataStreamClient connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The `RawDataStreamClient`::Connect() function sets up a unicast socket connection for the `RawDataStream` defined by the instance, and establishes a connection to the TCP server.

In the case of UDP, no connection is established. Incoming and outgoing packets are restricted to the specified address. The socket endpoints and attributes are specified in the manifest which is accessed through the InstanceSpecifer provided in the con-

structor. If TLS security protocol is configured for the socket connection, the TLS/DTLS connection shall be initialized here.⌋

[SWS_RDS_10514] Connect RawDataStreamServer to incoming connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamServer::WaitForConnection() function enables to setup the `RawDataStreamServer` instance for incoming connections.

For TCP the constructor marks the socket as ready to accept connection requests from a client, and WaitForConnection() waits to accept an incoming connection request.

In the case of UDP, no connection is established, and the operation shall return with no action.⌋

7.1.4.3 Shutdown of a RawDataStream connection

[SWS_RDS_10515] Shutdown RawDataStreamClient connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamClient::ShutDown() function closes the socket connection for the `RawDataStream` defined by the instance. Both the receiving and the sending part of the socket connection shall be shut down.

For TCP, the full-duplex connection shall be shut down disallowing further receptions and transmissions, before closing the socket.⌋

[SWS_RDS_10516] Shutdown RawDataStreamServer connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamServer::ShutDown() function closes the socket connection for the `RawDataStream` defined by the instance. Both the receiving and the sending part of the socket connection shall be shut down.

For TCP, the full-duplex connection shall be shut down disallowing further receptions and transmissions, before closing the socket.⌋

7.1.4.4 Read data from a RawDataStream connection

[SWS_RDS_10517] Read data from a RawDataStreamClient connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamClient::ReadData() function requests to read a number of bytes of data from the socket connection for the `RawDataStream` defined by the instance.

If Multicast Credentials are defined for the client, the data shall be read from the multicast socket created in the constructor (for 1:N use cases), otherwise the data shall be read from the unicast TCP socket connection set up in Connect() (for 1:1 TCP unicast

use case), or the unicast UDP socket created in the constructor (for 1:1 UDP unicast use case).

For efficiency, the zero-copy semantics of std::unique_ptr is used, which means that the ownership of the allocated memory of the read data is transferred to the application in the Read DataResult.data value.⌋

[SWS_RDS_10518] Read data from a RawDataStreamServer connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamServer::ReadData() function requests to read a number of bytes of data from the Unicast socket connection for the `RawDataStream` defined by the instance.

For efficiency, the zero-copy semantics of std::unique_ptr is used, which means that the ownership of the allocated memory of the read data is transferred to the application in the ReadDataResult.data value.⌋

7.1.4.5 Write data to a RawDataStream connection

[SWS_RDS_10519] Write data to a RawDataStreamClient connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamClient::WriteData() function requests to write of a number of bytes to the the socket connection for the `RawDataStream` defined by the instance (for 1:1 use cases).

If Multicast Credentials are defined for the client reading of data, a single socket can be used for both multicast reading and unicast writing (for use case 1:N UDP + 1:1 UDP, Server sends data via multicast, and a client sends control data via unicast). For efficiency, the zero-copy semantics of std::unique_ptr is used.⌋

[SWS_RDS_10520] Write data to a RawDataStreamServer connection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈The RawDataStreamClient::WriteData() function requests to write a number of bytes to the the socket connection for the `RawDataStream` defined by the instance.

If Remote Multicast Credentials are defined for the server, the data shall be written to the multicast socket created in the constructor (for 1:N use cases). Otherwise in case of TCP, the data shall be written to the unicast socket connection set up in WaitForConnection() (for 1:1 TCP unicast use case).

In case of UDP the data shall be written to the unicast socket created in the constructor (1:1 UDP unicast).

For efficiency, the zero-copy semantics of std::unique_ptr is used.⌋

7.2 Raw Data Streaming using IEEE1722 protocol (data link layer)

7.2.1 Raw Data Streaming Interface

The operations of the Raw Data Streaming interface using IEEE1722 protocol are synchronous and therefore performed as blocking operation calls.

The configuration of the `Raw Data Stream`s using IEEE1722 protocol is done by specifying parameters for the data link layer socket connections that shall be used for the communication via an `IEEE1722 stream`, using `RawDataStreamMapping` and `EthernetMacRawDataStreamMapping`. The model elements and the parameters are described in *TPS_ManifestSpecification* [8].

Secure communication can be achieved by using MACSec protocols in the middleware. Also access control imposed by the IAM can be applied for Raw Data Streams. All security functions (MACSec, IAM) are configurable in the deployment and mapping model of Raw Data Streaming Interface, see *TPS_ManifestSpecification* [8].

An application can use the Raw Data Streaming API both as a `IEEE1722 stream consumer` (consuming data from an `IEEE1722 stream`) or `IEEE1722 stream producer` (producing data for an `IEEE1722 stream`).

Figure 7.3 shows the logical view of the usage of Raw Data Streaming instances using IEEE1722 protocol.

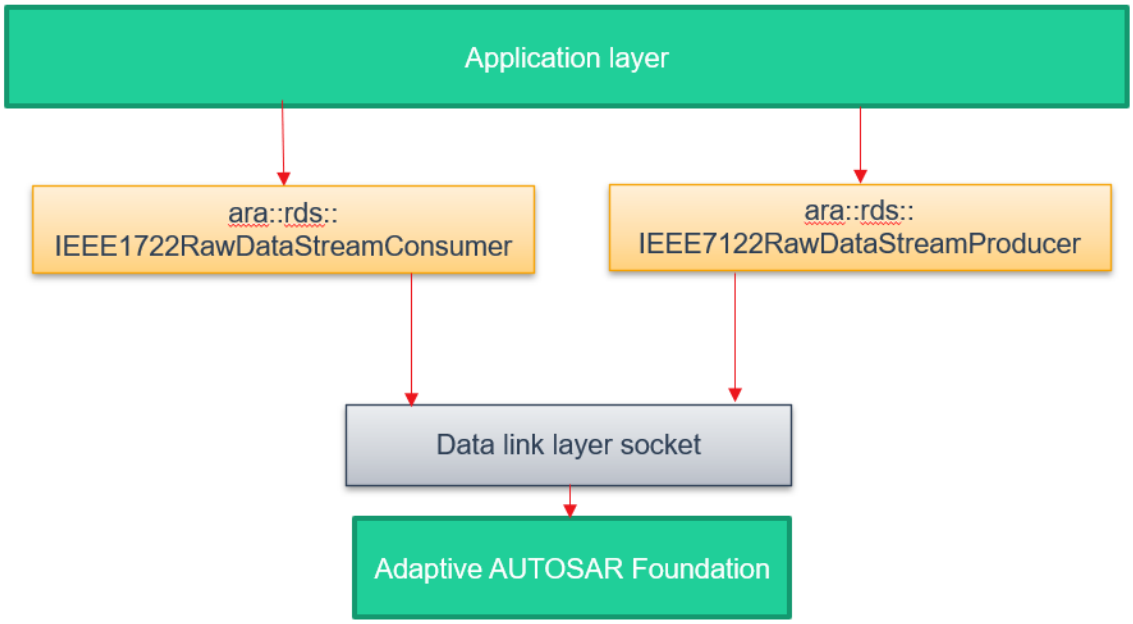

Figure 7.3: IEEE1722 Raw Data Stream Logical View.

7.2.1.1 Use cases

The `Raw data stream` interface supporting IEEE1722 protocol can be used in the following set-ups:

- `IEEE1722 stream consumer` which consumes data from an `IEEE1722 stream` via a data link layer socket connection.
- `IEEE1722 stream producer` which produces data to an `IEEE1722 stream` via a data link layer socket connection.

Data link layer sockets can be setup for a communication based on the IEEE1722 protocol. Currently the following use cases are supported:

- 1:1 communication mapping, where one `IEEE1722 stream producer` and one `IEEE1722 stream consumer` is connected to one `IEEE1722 stream` (e.g. video stream, where data is produced by one video device and data is consumed by one display).
- 1:n communication mapping, where one `IEEE1722 stream producer` and multiple `IEEE1722 stream consumer`s are connected to one `IEEE1722 stream` (e.g. audio stream, where data is produced by a audio device and data it consumed by multiple speakers).
- n:1 communication mapping, where multiple `IEEE1722 stream producer` and one `IEEE1722 stream consumer`s are connected to one `IEEE1722 stream` (e.g. sensor fusion stream, where data is produced by multiple radar sensors and a single data consumer fusions the data).
- n:m communication mapping, where multiple `IEEE1722 stream producer` and multiple `IEEE1722 stream consumer` are connected to one `IEEE1722 stream` (e.g. CAN frames from multiple buses of zone A are collected and transferred as IEEE1722 ACF encapsulated messages via an `IEEE1722 stream` (producer) and forwarded via an Ethernet switched network to multiple buses of zone B (consumer)).

7.2.2 Raw Data Streaming

`IEEE1722RawDataStream` communication is statically configured via the Deployment model as part of the Service Instance Manifest. The communication via IEEE1722 streams is stateless and driven unidirectionally between `IEEE1722RawDataStreamProducer` (source) and `IEEE1722RawDataStreamConsumer` (destination). An application, which need to consume data via an IEEE1722 stream, need to create an instances of `IEEE1722RawDataStreamConsumer`. An application, which need to produce data and forward it via an IEEE1722 stream, need to create an instances of `IEEE1722RawDataStreamProducer`. An instance for an `IEEE1722RawDataStreamConsumer` is created from the template class ara::rds::IEEE1722RawDataStreamConsumer (see [SWS_RDS_90311]), and an

instance for an `IEEE1722RawDataStreamProducer` is created from the template class ara::rds::IEEE1722RawDataStreamProducer (see [SWS_RDS_90322]).

The type for the template class could be one of the classes, which represent an IEEE1722 subtype that is supported by AUTOSAR (see Chapter 8.3):

- class ara::rds::IEEE1722Datagram61883_IIDC
- class ara::rds::IEEE1722DatagramAAF
- class ara::rds::IEEE1722DatagramTSCF
- class ara::rds::IEEE1722DatagramRVF
- class ara::rds::IEEE1722DatagramCRF
- class ara::rds::IEEE1722DatagramNTSCF

[SWS_RDS_10525] Supported types to create an instance of `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈The creation of an instance for an `IEEE1722RawDataStreamConsumer` or an `IEEE1722RawDataStreamProducer` shall consider to use the corresponding template classes ara::rds::IEEE1722RawDataStreamConsumer (see [SWS_RDS_90311]) or ara::rds::IEEE1722RawDataStreamProducer (see [SWS_RDS_90322]), where the given type shall be one of the following classes, that represents an specific IEEE1722 subtype. Any other type shall be prohibited:

- class ara::rds::IEEE1722Datagram61883_IIDC
- class ara::rds::IEEE1722DatagramAAF
- class ara::rds::IEEE1722DatagramTSCF
- class ara::rds::IEEE1722DatagramRVF
- class ara::rds::IEEE1722DatagramCRF
- class ara::rds::IEEE1722DatagramNTSCF

⌋

The IEEE1722 subtype classes reside as subordinated leaf classes of an class inheritance hierarchy. The hierarchy represents the IEEE1722 header formats and its different variants. Figure 7.4 shows the IEEE1722 subtype class inheritance hierarchy.

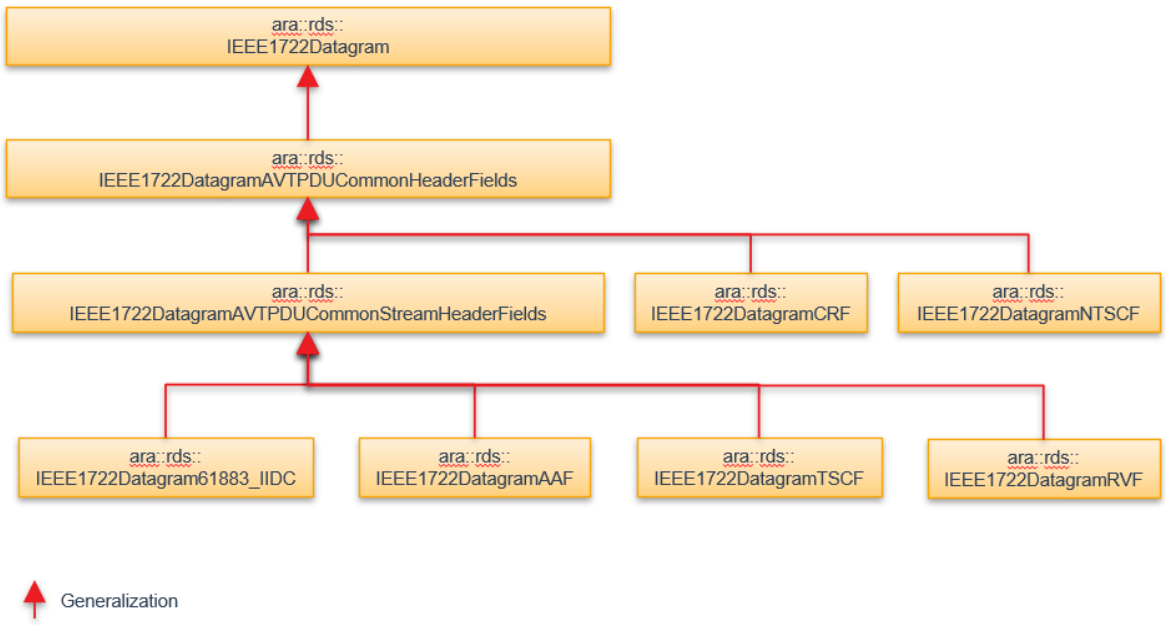

Figure 7.4: IEEE1722 subtype class inheritance hierarchy.

Each class within the IEEE1722 subtype class inheritance hierarchy contain specific IEEE1722 header fields as member variables (e.g. ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields::avtpStreamDataSubtype (see [SWS_RDS_90230])). The values of the member variables are used for different purposes:

- The values of the member variables are used to create an IEEE1722 header (a.k.a. AVTPDU-header) in case an application need to produce data (WriteData operation) via an instance of an `IEEE1722RawDataStreamProducer`. Some of the member variables are declared as optional. Optional member variables could be given by the application to be considered for the creation of an IEEE1722 header. This should support freedom for IEEE1722 related communiation at runtime. Therefore, the middleware has to determine the correct value for an IEEE1722 header field based on given values by application, configured value (derived from the Manifest) of the correspondng `IEEE1722RawDataStreamProducer`, a specified default value and the possibility to calculate missing values. Details of the expected behaviour for an IEEE1722 header creation is described in 7.2.4.2 “Produce IEEE1722 stream data”.
- The values of the member variables are used to perform an consistency check (a.k.a. AVTPDU-header inspection) in case an application need to consume data (ReadData operation) via an instance of an `IEEE1722RawDataStreamConsumer`. Details of the expected behaviour for an AVTPDU-header inspection is described in Chapter 7.2.4.1 “Consume IEEE1722 stream data”.

Each intance of an IEEE1722RawDataStreamConsumer and IEEE1722RawDataStreamProducer is identified by an unique InstanceSpecifier. The InstanceSpecifier is given to the constructor to create an specific instance of an IEEE1722RawDataStreamConsumer or IEEE1722RawDataStreamProducer. At creation of an specific instance at least the following points will be preformed:

- A data link layer socket connection is created which is based on the configuration of the Manifest that refer to the given InstanceSpecifier.
- The IEEE1722 related member variables (e.g. avtpStreamDataSubtype (see [SWS_RDS_90230])) of the created instance are set with the value configured in the Manifest that refer to the given InstanceSpecifier.

Thus, each created instance identified with the InstanceSpecifier reflect the configuration of the Manifest for a specific IEEE1722RawDataStreamConsumer or IEEE1722RawDataStreamProducer and refer to a created data link layer socket connection, using the artifacts specified in the mapped IEEE1722RawDataStreamConsumerMapping and IEEE1722RawDataStreamProducerMapping

For the C++ API reference of the Raw Data Stream interface using IEEE1722 protocol, see chapter 8.3 “Header: ara/rds/raw_data_stream_IEEE1722.h”.

[SWS_RDS_10526] Prepare a local connection to an IEEE1722RawDataStream

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈To prepare a local connection to an IEEE1722RawDataStream, an IEEE1722RawDataStreamConsumer or IEEE1722RawDataStreamProducer instance shall be created. The constructor shall create the necessary data link layer socket for IEEE1722RawDataStream Communication, using the artifacts specified in the mapped IEEE1722RawDataStreamConsumerMapping and IEEE1722RawDataStreamProducerMapping.⌋

As mentioned before, Data link layer socket connections using IEEE1722 protocol are statically configured in the Deployment model as part of the Service Instance Manifest, and used throughout the connected session for the IEEE1722RawDataStream communication. The following configuration elements can be specified on the Deployment model of each IEEE1722RawDataStreamConsumer or IEEE1722RawDataStreamProducer instance, identified through the InstanceSpecifier provided to the constructor.

[SWS_RDS_10527] Data link layer configuration of an IEEE1722RawDataStreamConsumer using IEEE1722 protocol

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each IEEE1722RawDataStreamConsumer instance identified by the InstanceSpecifier provided to the constructor, shall consider the following configuration elements specified on the Deployment model, if avaiable:

- Local Data Link Layer Socket: IEEE1722RawDataStreamConsumerMapping.localCommConnector and the corresponding EthernetCommunicationController referenced via EthernetCommunicationConnector.commController.
- IEEE1722 stream to consume data from: IEEE1722RawDataStreamConsumerMapping.ieee1722Stream.
- Socket Options: IEEE1722RawDataStreamConsumerMapping.socketOption.

⌋

[SWS_RDS_10528] Data link layer communication configuration of an `IEEE1722RawDataStreamConsumer` using IEEE1722 protocol

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈For each IEEE1722RawDataStreamConsumer instance identified by the InstanceSpecifier provided to the constructor, the following shall apply:

- If the IEEE1722TpConnection.destinationMacAddress referenced via IEEE1722RawDataStreamConsumerMapping.ieee1722Stream is set, then the data link layer socket shall consider the value as destination MAC address of expected received Ethernet frames.
- If the IEEE1722TpConnection.vlanPriority referenced via IEEE1722RawDataStreamConsumerMapping.ieee1722Stream is set, then the data link layer socket shall consider the value as VLAN priority of expected received Ethernet frames.

⌋

[SWS_RDS_10529] Data link layer configuration of an `IEEE1722RawDataStreamProducer` using IEEE1722 protocol

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each IEEE1722RawDataStreamProducer instance identified by the InstanceSpecifier provided to the constructor, shall consider the following configuration elements specified on the Deployment model, if avaiable:

- Local Data Link Layer Socket: IEEE1722RawDataStreamProducerMapping.localCommConnector and the corresponding EthernetCommunicationController referenced via EthernetCommunicationConnector.commController.
- IEEE1722 stream to produce data to: IEEE1722RawDataStreamProducerMapping.ieee1722Stream.
- Socket Options: IEEE1722RawDataStreamProducerMapping.socketOption.

⌋

[SWS_RDS_10530] Data link layer communication configuration of an `IEEE1722RawDataStreamProducer` using IEEE1722 protocol

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈For each `IEEE1722RawDataStreamProducer` instance identified by the InstanceSpecifier provided to the constructor, the following shall apply:

- If the `EthernetCommunicationConnector.maximumTransmissionUnit` of the referenced `IEEE1722RawDataStreamProducerMapping.localCommConnector` is set, then the data link layer socket shall consider the value as maximum transmission unit for transmitted Ethernet frames.
- If the `IEEE1722TpConnection.destinationMacAddress` referenced via `IEEE1722RawDataStreamProducerMapping.ieee1722Stream` is set, then the data link layer socket shall consider the value as destination MAC address for transmitted Ethernet frames.
- If the `IEEE1722TpConnection.vlanPriority` referenced via `IEEE1722RawDataStreamProducerMapping.ieee1722Stream` is set, then the data link layer socket shall consider the value as VLAN priority for transmitted Ethernet frames.

⌋

[SWS_RDS_10531] Socket Options configuration

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈For both `IEEE1722RawDataStreamConsumer`s and `IEEE1722RawDataStreamProducer`s a list of data link layer socket options can be defined in the attribute `socketOption` to be applied to the created data link layer sockets . The options shall be specified as a list of strings. The accepted values are platform specific and shall be documented by the vendor.⌋

An example of `socketOption` definition is to provide a series of "option", "value" pairs for data link layer socket options, e.g.: ["Socket_Type", "SOCK_PACKET"]

[SWS_RDS_10532] Derive IEEE1722 protocol configuration at creation of an `IEEE1722RawDataStreamConsumer` instance

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each `IEEE1722RawDataStreamConsumer` instance identified by the InstanceSpecifier provided to the constructor, shall derive the IEEE1722 configuration that belongs to the `IEEE1722TpConnection` which is referenced via `IEEE1722RawDataStreamConsumerMapping.ieee1722Stream` specified on the Deployment model. At creation of this `IEEE1722RawDataStreamConsumer` in-

stance, the member variables of this instance shall be set to the corresponding configuration values that belongs to the referenced`IEEE1722TpConnection`.⌋

[SWS_RDS_10533] Derive IEEE1722 protocol configuration at creation of an `IEEE1722RawDataStreamProducer` instance

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each `IEEE1722RawDataStreamProducer` instance identified by the InstanceSpecifier provided to the constructor, shall derive the IEEE1722 configuration that belongs to the `IEEE1722TpConnection` which referenced via `IEEE1722RawDataStreamProducerMapping` in the role of `IEEE1722RawDataStreamProducerMapping.ieee1722Stream` specified on the Deployment model. At creation of this `IEEE1722RawDataStreamProducer` instance, the member variables of this instance shall be set to the corresponding configuration values that belongs to the referenced `IEEE1722TpConnection`.⌋

[SWS_RDS_10534] Error handling if IEEE1722 protocol configuration is derived

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each invocation of the Create operation either of an `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` shall derive the configured IEEE1722 configuration with respect to [SWS_RDS_10532] and [SWS_RDS_10533]. If the the given type of the `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` do not match to the corresponding `IEEE1722TpConnection`, then it shall be handled as an InstanceSpecifierMappingIntegrityViolation.⌋

The functionality of a `IEEE1722RawDataStream` for an `IEEE1722RawDataStreamConsumer` communication is realized in these three operations: Connect, Shutdown, and ReadData. The functionality of a `IEEE1722RawDataStream` for an `IEEE1722RawDataStreamProducer` communication is realized in these three operations: Connect, Shutdown and WriteData.

[SWS_RDS_10535] Establish local communication connection to an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each invocation of the Connect operation either of an `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` instance shall request to establish a local communication connection to a data link layer socket. The data link layer socket created for an specific `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer`⌋

[SWS_RDS_10536] Error handling for establishing a local communication connection to an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If the Connect operation either of an `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` instance is invoked and the communication connection of the corresponding data link layer socket has already been established, then the operation shall do nothing and return with kStreamAlreadyConnected.⌋

[SWS_RDS_10537] Shutdown local communication connection to an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each invocation of the Shutdown operation of an `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` instance shall request to disconnect from a local communication connection via the corresponding data link layer socket.⌋

[SWS_RDS_10538] Error handling for shutdown a local communication connection to an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If the Shutdown operation either of an `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` instance is invoked and the communication connection of the corresponding data link layer socket has already been disconnected, then the operation shall do nothing and return with kStreamNotConnected.⌋

[SWS_RDS_10539] Destructor behavior for local communication connection to an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If the destructor either of an `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer` instance is executed where the local communication connection to the data link layer is still connected, then the destructor shall perform a Shutdown operation internally before the object is destroyed.⌋

7.2.3 Security

7.2.3.1 Access Control via IAM

[SWS_RDS_10540] Restrictions on using `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_AP_00158

⌈If a `Process` calls the

- the `IEEE1722RawDataStreamConsumer::Create()` constructor (see [SWS_RDS_90312])

or

- the `IEEE1722RawDataStreamProducer::Create()` constructor (see [SWS_RDS_90323])

providing an InstanceSpecifier that does not identify a `RPortPrototype` referenced by a `RawDataStreamMapping` in the role `portPrototype`, that is mapped to the requesting `Process` in the role `process`, then this shall be treated as a violation.⌋

7.2.3.2 Secure Communication

`IEEE1722RawDataStream` communication could be secured by using MACSec protocol on the corresponding data link layer. This machnism is out of scope of the `Raw data stream` functional cluster and need to be established by the system.

7.2.4 Raw Data Streaming Interfaces

7.2.4.1 Consume IEEE1722 stream data

[SWS_RDS_10541] Read data from an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each invocation of the ReadData operation of an `IEEE1722RawDataStreamConsumer` instance shall request to read at most number of datagrams from the connected `IEEE1722RawDataStream` given with argument maxNumberOfDatagrams. The amount of read datagram(s) shall be moved to a buffer returned as result from the function. Only those datagrams shall be returned, where the AVTPDU-header inspection was successfully finished.⌋

The AVTPDU-header inspection is described in 7.2.4.1.1 “AVTPDU-header inspection”.

Note: The datagram(s) are returned as ara::core::Vector configured as IEEE1722 subtype (e.g. ara::rds::IEEE1722AAFDatagram). The properties (e.g. number of returned datagrams) of the returned vector can be accessed by public member functions provided by ara::core::Vector class (e.g. <vector instance>.size)

7.2.4.1.1 AVTPDU-header inspection

Inspection of the AVTPDU-header include consistency of the received AVTPDU-header fields compared to the corresponding configuration of the `IEEE1722RawDataStreamConsumer` instance. This is performed when the

application consumes data via an `IEEE1722RawDataStreamConsumer` instance by invoking ReadData operation. The inspection of AVTPDU-header considers the AVTPDU-header format specified by [1, IEEE1722]. [1, IEEE1722] specify 4 different formats of the AVTPDU-header format: AVTPDU-common-header, AVTPDU-common-stream-header, AVTPDU-common-control-header and AVTPDU-alternative-header. Please note, AVTPDU-common-control-header fields are not considered, since the supported `IEEE1722TpStream`s in AUTOSAR do not use the AVTPDU-common-control-header format.

[SWS_RDS_10542] Inspection of AVTPDU-common-header fields

Status: DRAFT
Upstream requirements: FO_RS_IEEE1722_00002

⌈If an AVTPDU-header inspection is performed, then the process shall inspect the AVTPDU-common-header fields according the format specified by [1, IEEE1722] and consider the following consistency checks:

- If value of subtype field of the inspected AVTPDU-header match to the configured subtype of the `IEEE1722RawDataStreamConsumer` instance (see [SWS_RDS_90230]), then the AVTPDU-header inspection shall proceed. Otherwise the AVTPDU-header inspection shall be aborted and the affected datagram shall be discarded.
- If version field of the inspected AVTPDU-header match to the configured subtype of the `IEEE1722RawDataStreamConsumer` instance (see [SWS_RDS_90232]), then the AVTPDU-header inspection shall proceed. Otherwise the AVTPDU-header inspection shall abort and discard the affected datagram.

⌋

[SWS_RDS_10543] Inspection of AVTPDU-common-stream-header fields

Status: DRAFT
Upstream requirements: FO_RS_IEEE1722_00002

⌈If an AVTPDU-header inspection is performed, then the process shall inspect the AVTPDU-common-stream-header fields according the format specified by [1, IEEE1722] and consider the following consistency checks:

- If avtp_timestamp field of the inspected AVTPDU-header represents a time value that is greater than the time value of the current synchronized global time, then AVTPDU-header inspection shall proceed. Otherwise the AVTPDU-header inspection shall be aborted, a presentation time violation logged and the affected datagram discarded.

⌋

[SWS_RDS_10544] Consistency checks for stream header field values of AVTP common-stream-header format, AVTP stream data subtype CRF and NTSCF

Status: DRAFT
Upstream requirements: FO_RS_IEEE1722_00002

⌈The AVTPDU-header inspection shall inspect for `IEEE1722 stream`s with either the following AVTPDU-stream properties:

- `IEEE1722 stream` with AVTP common-stream-header format
- `IEEE1722 stream` of AVTP stream data subtype `CRF`
- `IEEE1722 stream` of AVTP stream data subtype `NTSCF`

the header fields according the format specified by [1, IEEE1722] and consider the following consistency checks:

- If sequence_num (sequence number) increase continuously and warp around at reaching maximum value, then the AVTPDU-header inspection shall proceed. Otherwise the AVTPDU-header inspection shall proceed and the sequence number mismatch shall be logged.
- If stream id field of the inspected AVTPDU-header match to the configured composite of the `IEEE1722TpStreamIdMacAddress` and `IEEE1722TpStreamIdUniquePart` at the `IEEE1722RawDataStreamConsumer`, then AVTPDU-header inspection shall proceed. Otherwise the AVTPDU-header inspection shall be aborted for this datagram and stream id mismatch shall be logged.

⌋

7.2.4.2 Produce IEEE1722 stream data

An application which produce data and need to transfer the data via an IEEE1722 stream has to create an `IEEE1722RawDataStreamProducer` instance. An `IEEE1722RawDataStreamProducer` instance is created via the template class ara::rds::IEEE1722RawDataStreamProducer, where the given type represents an specific IEEE1722 subtype. This `IEEE1722RawDataStreamProducer` instance represents the configuration of the Manifest for a specific `IEEE1722RawDataStreamProducer` identified with the InstanceSpecifier. The configuration values are derived to member variables of the `IEEE1722RawDataStreamProducer` instance when the instance is created. An application has to provide the data as ara::core::Vector, where each element of the Vector holds the information for excatly one datagram (e.g. stream data length, stream data payload ... a.s.o.). The type of a datagram need to match to the type of the `IEEE1722RawDataStreamProducer` instance, otherwise the datagram is skipped and not transfered as IEEE1722 stream. Therefore an application has to create an local instance of a class the corresponds to the IEEE1722 subtype for one datagram and set all values for non-optional member variables. Optional member

variables could be set by the applicaton, if needed. All datagrams which need to be transfered via an IEEE1722 stream, need to be provided as ara::core::Vector, where each element contains information (mandatory and may optional IEEE1722 header field values) of one datagram, to the WriteData operation of the affected `IEEE1722RawDataStreamProducer` instance. Within the WriteData operation a complete IEEE1722 datagram (IEEE1722 header and IEEE1722 payload) of specific IEEE1722 subtype (defined by [1, IEEE1722]) is created. The creation process for the IEEE1722 header need to determine each mandatory IEEE1722 header field value by considering the following points:

- Mandatory header field values given by the application are checked, and if valid, transfererd to IEEE1722 header field values of the created IEEE1722 stream subtype. Otherwise this datagram is skipped.
- Optional header field valus given by the application are checked, and if valid, transfered to IEEE1722 header field values of the created IEEE1722 stream subtype. Otherwise check if an configured value of within the `IEEE1722RawDataStreamProducer` instance is avialable, and if available, use this configufred value. Otherwise check, if an default value is specified or the value can be calculated, and if so, transfer the determined value as IEEE1722 header field value to the created IEEE1722 stream subtype. Otherwise this datagram is skipped.

[SWS_RDS_10545] Write data to an `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈Each invocation of the WriteData operation of an `IEEE1722RawDataStreamProducer` instance shall request to write a number of datagrams to the connected `IEEE1722RawDataStream` by iterating across the given ara::core::Vector with contained datagrams and perform for each datagram the following actions:

- Perform a check of the given IEEE1722 header field information with respect to the configured IEEE1722 stream subtype and consider the following points
  - If a mandatory header field (according to [1, IEEE1722]) provided via an instance of the configured class that represents the IEEE1722 subtype is missing and neither default value is specified nor the value can be calculated, then skip this datagram, log the absence of a mandatory header field value and proceed with the next available datagram. Otherwise consider the default or calculated value for this missing header field and proceed with the check of the IEEE1722 header field information.
  - If a mandatory header field (according to [1, IEEE1722]) provided via an instance of the configured class that represents the IEEE1722 subtype is available and the value exceed the valid value range, then skip this datagram, log a value range violation and proceed with the next available datagram. Otherwise proceed with the check of the IEEE1722 header field information.

- If the check for the IEEE1722 header field information was successfully finished, then create an IEEE1722 header according to [1, IEEE1722]) with respect to configured IEEE1722 stream subtype and concatinate the given data as payload.
- Send out the constructed IEEE1722 datagram on the corresponding data link socket.

As last step return the number of send datagrams.⌋

Note:
The IEEE1722 subtypes are described in 8.3 “Header: ara/rds/raw_data_stream_-IEEE1722.h”.
The creation of the IEEE1722 header is described in 7.2.4.2.1 “AVTPDU-header creation”).

7.2.4.2.1 AVTPDU-header creation

Creation of the AVTPDU-header include consistency / completeness of the given AVTPDU-header field values provided by the application, compared to the corresponding configuration of the `IEEE1722RawDataStreamProducer` instance. This is performed when the application produces data via an `IEEE1722RawDataStreamProdcuer` instance by invoking WriteData operation. The creation of AVTPDU-header is based on the AVTPDU-header format specified by [1, IEEE1722]. [1, IEEE1722] specify 4 different formats of the AVTPDU-header format: AVTPDU-common-header, AVTPDU-common-stream-header, AVTPDU-common-control-header and AVTPDU-alternative-header. Some of the header fields, which need a specific treatment and shared between the different formats (e.g. stream id), are embraced in sub-chapter Chapter 7.2.4.2.1.1. The subsequential sub-chapters describe how to set the header field values of AVTPDU-common-header, AVTPDU-common-stream-header, AVTPDU-alternative-header and the AVTP subtype specific format. Please note, AVTPDU-common-control-header fields are not considered, since the supported `IEEE1722 stream`s in AUTOSAR do not use the AVTPDU-common-control-header format.

7.2.4.2.1.1 Treatment of shared AVTPDU-header fields

[SWS_RDS_10546] Preparation of IEEE1722 header field value "sequence number"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈The `IEEE1722RawDataStreamConsumer` instance or `IEEE1722RawDataStreamProducer` instance shall maintain for each configured `IEEE1722 stream` with either the following AVTPDU-stream properties:

- `IEEE1722 stream` with AVTP common-stream-header format.

- IEEE1722 stream of AVTP stream data subtype CRF.
- IEEE1722 stream of AVTP stream data subtype NTSCF.

a separate sequence number and consider the following points:

- The sequence number of an particular IEEE1722 stream shall be increased with 01_{16} on each request for header creation.
- If the sequence number reaches the maximum value, then it should re-start with value 00_{16}.

⌋

[SWS_RDS_10547] Preparation of IEEE1722 header field value "stream id"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream with either the following AVTPDU-stream properties:

- IEEE1722 stream with AVTP common-stream-header format.
- IEEE1722 stream of AVTP stream data subtype CRF.
- IEEE1722 stream of AVTP stream data subtype NTSCF.

then the following points for creation of the IEEE1722 stream id shall be considered:

- If the MAC address is provided by the application and qualified as valid, then the provided MAC address shall be used. Otherwise the configured MAC address (IEEE1722TpConnection.destinationMacAddress or IEEE1722TpConnection.macAddressStreamId referenced via IEEE1722RawDataStreamConsumerMapping.ieee1722Stream) of the IEEE1722RawDataStreamConsumer instance or IEEE1722RawDataStreamProducer instance.
- If the unique id is provided by the application and qualified as valid, then the provided unique id shall be used. Otherwise the process shall use the configured unique id (IEEE1722TpConnection.uniqueStreamId referenced via IEEE1722RawDataStreamConsumerMapping.ieee1722Stream) of the processed IEEE1722 stream.

⌋

[SWS_RDS_10548] Determination of IEEE1722 header field value "time uncertain"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈An IEEE1722RawDataStreamConsumer instance or IEEE1722RawDataStreamProducer instance shall use the GlobalTimeDo-

main referenced by IEEE1722TpConnection.globalTimeDomain to determine the tu header field value with respect to the following rules:

- If the synchronization state of the GlobalTimeDomain is uncertain, then tu (time uncertain) header field shall be set to 1.
- If the GlobalTimeDomain is synchronized, then set the tu (time uncertain) header field value to 0.

⌋

[SWS_RDS_10549] Handling if length of AVTPDU-header and AVTP-payload exceeds maxiumn transmission unit

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream and the accumulated length of AVTPDU-header and AVTP-payload exceed the MTU (Maximum Transmission Unit) of the corresponding data link layer socket (see [SWS_RDS_10527]), then the AVTPDU-header creation shall be aborted.⌋

7.2.4.2.1.2 AVTPDU-common-header fields

The AVTPDU-common-header format is shared between all AVTP stream data subtypes. This chapter describe how to create and set values of the AVTPDU-common-header fields according to [1, IEEE1722] chapter "4.4.3 AVTPDU common header format".

[SWS_RDS_10550] Determination of IEEE1722 header field value "AVTP stream data subtype"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream, then the subtype field shall be set with the AVTP stream data subtype according to [1, IEEE1722] chapter "4.4.3 AVTPDU common header format":

- If the processed IEEE1722 stream has CRF configured (see [SWS_RDS_90230]), then the subtype field shall be set to AVTP stream data subtype value 04_{16}.
- If the processed IEEE1722 stream has AAF configured (see [SWS_RDS_90230]), then the subtype field shall be set to AVTP stream subtype value 02_{16}.
- If the processed IEEE1722 stream has IEC61883_IIDC configured (see [SWS_RDS_90230]), then the subtype field shall be set to AVTP stream subtype value 00_{16}.

- If the processed IEEE1722 stream has RVF configured (see [SWS_RDS_90230]), then the subtype field shall be set to AVTP stream subtype value 07_{16}.
- If the processed IEEE1722 stream has NTSCF configured (see [SWS_RDS_90230]), then the subtype field shall be set to AVTP stream subtype value 82_{16}.
- If the processed IEEE1722 stream has TSCF configured (see [SWS_RDS_90230]), then the subtype field shall be set to AVTP stream subtype value 05_{16}.

⌋

[SWS_RDS_10551] Setting of IEEE1722 header field value "version"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream, then the process shall set the version field (see [1, IEEE1722] chapter "4.4.3 AVTPDU common header format") to the configured value of the IEEE1722 stream (see [SWS_RDS_90232])⌋

Note: The value for the h (header specific) field is specified in Chapter 7.2.4.2.1.3 and Chapter 7.2.4.2.1.4

7.2.4.2.1.3 AVTPDU-common-stream-header fields

[SWS_RDS_10552] Determination of IEEE1722 header field value "media clock restart"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream with an AVTPDU-common-stream-header format and media clock restart value is provided by the application and qualified as valid, then the process shall set the mr (media clock restart) field to the given value (see [1, IEEE1722] chapter "4.4.4 AVTPDU common stream header"). Otherwise set this header field to zero.⌋

[SWS_RDS_10553] Setting of IEEE1722 header field value "stream id valid"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream with an AVTPDU-common-stream-header format, then the process shall set sv (stream_id valid) field to 1 (see [1, IEEE1722] chapter "4.4.4.2 sv (stream_id valid) field")⌋

[SWS_RDS_10554] Determination of IEEE1722 header field value "sequence number"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with an AVTPDU-common-stream-header format, then the process shall set the sequence_num (sequence number) field to the current value with respect to (see [1, IEEE1722] chapter "4.4.4 AVTPDU common stream header") and [SWS_RDS_10546])⌋

[SWS_RDS_10555] Setting of IEEE1722 header field value "timestamp uncertain"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with an AVTPDU-common-stream-header format, then the process set the tu (timestamp uncertain) field (see [1, IEEE1722] chapter "4.4.4 AVTPDU common stream header") to the determined value as specified with [SWS_RDS_10548].⌋

[SWS_RDS_10556] Creation of IEEE1722 header field value "stream id"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with an AVTPDU-common-stream-header format, then the process shall construct a stream id with respect to [SWS_RDS_10547] and set the stream_id (stream id) field of the AVTPDU-header (see [1, IEEE1722] chapter "4.4.4 AVTPDU common stream header") to the constructed stream id.⌋

[SWS_RDS_10557] Determination of IEEE1722 header field value "avtp timestamp"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with an AVTPDU-common-stream-header format and the avtp timestamp value is provided by the application and qualified as valid, then the process shall set the avtp_timestamp (avtp timestamp) field of the AVTPDU-header (see [1, IEEE1722] chapter "4.4.4 AVTPDU common stream header") to available avtp timestamp value. Otherwise the process shall calculate avtp timestamp according the following equation and set the avtp_timestamp (avtp timestamp) field of the AVTPDU-headerfield to the calcuated presentation time:

$$T_{\text{presentation\_time}} = T_{\text{current\_synchronized\_globaltime}} + T_{\text{maxTransitTime}} \tag{7.1}$$

`maxTransitTime` shall be determined by the `IEEE1722TpAvConnection.maxTransitTime` or `IEEE1722TpAcfConnection.acfMaxTransitTime` referenced via `IEEE1722RawDataStreamProducerMapping.ieee1722Stream`) of the `IEEE1722RawDataStreamProducer` instance.⌋

[SWS_RDS_10558] Setting of IEEE1722 header field value "stream data length"

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with an AVTPDU-common-stream-header format, then the process shall set stream_data_length (stream data length) field of the AVTP-payload (see [1, IEEE1722] chapter "4.4.4 AVTPDU common stream header") to length in bytes given with the instance of the datagram (e.g. instance of class IEEE1722DatagramAAF).⌋

7.2.4.2.1.4 AVTPDU-alternative-header fields

The AVTPDU-alternative-header fields are AVTP stream data subtype specific and described in the according subchapters for `IEEE1722 stream` of AVTP stream data subtype `CRF` and `NTSCF`.

7.2.4.2.1.5 61883_IIDC-header fields

This chapter describe how to create values which are specific for AVTP stream data subtype "61883_IIDC" (IEC 61883/IIDC format) according to [1, IEEE1722] chapter "5. IEC 61883/IIDC Format".

[SWS_RDS_10559] Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field is set to 0

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `61883_IIDC` and the configured tag field (see [SWS_RDS_90244] of the `IEEE1722RawDataStreamProducer` instance is set to 0, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "5.2 IEC 61883/IIDC stream data encapsulation" in addition to the AVTPDU-common-header fields and AVTPDU-common-stream-header fields:

- Set gv (gateway_info valid) field to zero.
- Set gateway_info field to zero.
- Set tag field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90244]).
- Set channel field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90245]).
- Set tcode (type code) field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90246]).

- Set sy field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90247]).

⌋

Note for [SWS_RDS_10559]: Refer to Chapter 7.2.4.2.1.2 for details on AVTPDU-common-header fields and to Chapter 7.2.4.2.1.3 for details on AVTPDU-common-stream-header fields.

[SWS_RDS_10560] Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field is set to 1

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `61883_IIDC` and the configured tag field (see [SWS_RDS_90244]) of the `IEEE1722RawDataStreamProducer` instance is set to 1, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "5.4.3 IEC 61883 CIP header encapsulation" in addition to [SWS_RDS_10559]:

- Set qi_1 (quadlet indicator) field to 00_2.
- Set SID (source identifier) field to 63_{10}.
- Set DBS (data block size) field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90250]).
- Set FN (fraction number) field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90251]).
- Set QPC (quadlet padding count) field to value provided by the application. If value is not available or invalid, set the value to 0.
- Set SPH (source packet header) field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90253]).
- Set DBC (data block count) field to value provided by the application. If value is not available or invalid, set the value to 0.
- Set qi_2 (quadlet indicator) field to 10_2.
- Set FMT (stream format) field to configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90256]).

⌋

Note: AUTOSAR do not support AVTP gateway function

[SWS_RDS_10561] Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field is set to 1 and configured SPH field is set to 0

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `61883_IIDC`, the configured tag field (see [SWS_RDS_90244]) is set to 1 and the configured SPH field ([SWS_RDS_90257]) is set to 0 of the `IEEE1722RawDataStreamProducer` instance, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "5.4.4 IEC 61883 (SPH = 0) encapsulation" in addition to [SWS_RDS_10560]:

- Set tv (avtp_timestamp valid) field to value provided via meta data. If value is not available or invalid, set the value to 0.
- Set avtp_timestamp field according to [SWS_RDS_10557].
- Set FDF (format dependent field) field to value provided by the application. If value is not available or invalid, set the value to 0.
- Set SYT (synchronization timing) field to $FFFF_{16}$.
- Set cip_no_sph_payload field to IEEE1722Datagram::data given with the instance of the datagram to be transmitted.

⌋

[SWS_RDS_10562] Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field and configured SPH field are set to 1

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `61883_IIDC`, the configured tag field (see [SWS_RDS_90244]) is set to 1 and the configured SPH field ([SWS_RDS_90257]) is set to 1 of the `IEEE1722RawDataStreamProducer` instance, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "5.4.4 IEC 61883 (SPH = 1) encapsulation" in addition to [SWS_RDS_10560]:

- Set tv (avtp_timestamp valid) field to value 0.
- Set FDF (format dependent field) field to value provided by the application. If value is not available or invalid, set the value to 0.
- Set cip_with_sph_payload field to IEEE1722Datagram::data given with the instance of the datagram to be transmitted.

⌋

Note: The avtp_source_packet_header_timestamp field is included in the cip_with_sph_payload. The cip_with_sph_payload could include multiple source packets.

7.2.4.2.1.6 AAF-header fields

This chapter describe how to create values which are specific for AVTP stream data subtype "AAF" (AVTP Audio Format) according to [1, IEEE1722] chapter "7. AVTP Audio Format".

[SWS_RDS_10563] Setting of IEEE1722 subtype stream AAF header field values

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `AAF`, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "7.2 AAF common stream data encapsulation" in addition to the AVTPDU-common-header fields and AVTPDU-common-stream-header fields:

- Set format field to value of configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90262]).
- Set sp (sparse timestamp) field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90263]).
- Set evt field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90264]).

⌋

Note for [SWS_RDS_10563]: Refer to Chapter 7.2.4.2.1.2 for details on AVTPDU-common-header fields and to Chapter 7.2.4.2.1.3 for details on AVTPDU-common-stream-header fields.

[SWS_RDS_10564] Setting of IEEE1722 subtype stream AAF header field values if AAF AVTP format PCM is indicated

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `AAF` and the configured Format field (see [SWS_RDS_90262]) of the `IEEE1722RawDataStreamProducer` instance is set to a value that indicates AAF AVTP format PCM, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "7.3 AAF PCM stream data encapsulation" additional to [SWS_RDS_10563]:

- Set nsr (nominal sample rate) field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90265]).
- Set channels_per_frame field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90266]).
- Set bit_depth field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90267]).

- Set pcm_data_payload field to data given with the instance of the datagram (e.g. instance of class IEEE1722DatagramAAF) to be transmitted.

⌋

[SWS_RDS_10565] Setting of IEEE1722 subtype stream AAF header field values if AAF AVTP format AES3 is indicated

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `AAF` and the configured Format field (see [SWS_RDS_90262]) of the `IEEE1722RawDataStreamProducer` instance is set to a value that indicates AAF AVTP format AES3, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "7.3 AAF PCM stream data encapsulation" additional to [SWS_RDS_10563]:

- Set nfr (nominal frame rate) field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90268]).
- Set streams_per_frame field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90269]).
- Set aes3_data_type_h field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90270]).
- Set aes3_dt_ref field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90267]).
- Set aes3_data_type_l field to the configured value the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90271]).

⌋

7.2.4.2.1.7 ACF-header fields

This chapter describe how to create values which are specific for AVTP stream data subtype "ACF" (AVTP Control Format) according to [1, IEEE1722] chapter "9. AVTP Control Format".

Note:

- AUTOSAR do not support stream reservation protocol (SRP), but due to [1, IEEE1722] chapter "4.4.4.2 sv (stream_id valid) field" the sv field is always set to 1 (see [SWS_RDS_10551]).
- A `IEEE1722 stream` of subtype `NTSCF` (non time synchronous control format) or `TSCF` (time synchronous control format) transport bus frams as ACF-messages with the ACF-message-payload. The ACF-messsage-payload can carry one or more arbitrary ACF-messages. The creation of ACF-messages and the resulting

ACF-message-payload according to [1, IEEE1722] chapter "9.4 ACF messages" is out of scope for the Raw Data Stream functional cluster. The ACF-message-payload has to be provided by an application.

[SWS_RDS_10566] Setting of IEEE1722 subtype stream NTSCF header field values

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream with AVTP stream data subtype set to NTSCF, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "9.2 Non-Time-Synchronous Control Format header" in addition to the AVTPDU-common-header fields:

- Set sv (stream_id valid) field to 01_{16} (see [SWS_RDS_10551]).
- Set ntscf_data_length field to the accumlated length of all ACF-messages transmitted as AVTPDU-payload of this IEEE1722TpStream.
- Set acf_payload_data field to the data of given with the instance of the datagram (i.e. instance of class IEEE1722DatagramNTSCF) to be transmitted.

⌋

Note for [SWS_RDS_10566]: Refer to Chapter 7.2.4.2.1.2 for details on AVTPDU-common-header fields.

[SWS_RDS_10567] Setting of IEEE1722 subtype stream TSCF header field values

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an IEEE1722 stream with AVTP stream data subtype set to TSCF, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "9.3 Time-Synchronous Control Format header" in addition to the AVTPDU-common-header fields and AVTPDU-common-stream-header fields:

- Set stream_data_length to the length in bytes given with the instance of the datagram (i.e. instance of class IEEE1722DatagramTSCF).
- Set acf_payload_data field to the data given with the instance of the datagram (e.g. instance of class IEEE1722DatagramTSCF) to be transmitted.

⌋

Note for [SWS_RDS_10567]: Refer to Chapter 7.2.4.2.1.2 for details on AVTPDU-common-header fields and to Chapter 7.2.4.2.1.3 for details on AVTPDU-common-stream-header fields.

7.2.4.2.1.8 CRF-header fields

This chapter describe how to create values which are specific for AVTP stream data subtype "CRF" (Clock Reference Format) according to [1, IEEE1722] chapter "10. Clock Reference Format".

[SWS_RDS_10568] Setting of IEEE1722 subtype stream CRF header field values

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `CRF`, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "10.4 Clock Reference Format Data encapsulation" in addition to the AVTPDU-common-header fields:

- Set sv (stream_id valid) field to value provided by the application and qualified as valid. Otherwise set this field to 0.
- Set mr (media clock reset) field to value provided by the application and qualified as valid. Otherwise set this field to 0.
- Set fs (frame sync) field to value provided by the application and qualified as valid. Otherwise set this field to 0.
- Set tu (timestamp uncertain) field to value determined as specified with [SWS_RDS_10548]. Otherwise set this field to 1.
- Set sequence_num field to value determined as specified with [SWS_RDS_10546].
- Set type field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90298]).
- Set stream_id field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_10547]).
- Set pull field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90301]).
- Set base_frequency field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90302]).
- Set crf_data_length field to length in bytes given with the instance of the datagram (i.e. instance of class IEEE1722DatagramCRF).
- Set timestamp_interval field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90303]).
- Set crf_data field to data given with the instance of the datagram (i.e. instance of class IEEE1722DatagramCRF) to be transmitted.

⌋

Note to [SWS_RDS_10568]:

- The remaining fields specified in [1, IEEE1722] chapter "10.4.13 crf_data field" reside in the crf_data field, which is provided within the CRF-payload by an application. The following fields are out of scope for the `Raw Data Stream` functional cluster: User-specified type, Audio sample type, Video frame sync type, Video line sync type, Machine cycle type.
- Refer to Chapter 7.2.4.2.1.2 for details on AVTPDU-common-header fields.

7.2.4.2.1.9 RVF-header fields

This chapter describe how to create values which are specific for AVTP stream data subtype "RVF" (Raw Video Format) according to [1, IEEE1722] chapter "10. Clock Reference Format".

[SWS_RDS_10569] Setting of IEEE1722 subtype stream RVF header field values

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If an AVTPDU-header is created for an `IEEE1722 stream` with AVTP stream data subtype set to `RVF`, then the process shall set the values for the specific header fields to the following values according to [1, IEEE1722] chapter "12.2 Raw Video Stream data encapsulation" in addition to the AVTPDU-common-header fields and AVTPDU-common-stream-header fields:

- Set active_pixels field to value provided by the application. If value is not available or invalid, set this field to 0.
- Set total_lines field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90278]).
- Set ap (active pixels) field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90279]).
- Set f (field) field to the value provided by the application. If value is not available or invalid, set this field to 0.
- Set ef (end frame) field to the value provided by the application. If value is not available or invalid, set this field to 0.
- Set evt field to configured value to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90282]).
- Set pd (pull-down) field to the value provided by the application. If value is not available or invalid, set this field to 0.
- Set i (interlaced) field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90284]).

- Set pixel_depth field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90285]).
- Set pixel_format field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90286]).
- Set frame_rate field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90287]).
- Set colorspace field to the configured value of the `IEEE1722RawDataStreamProducer` instance (see [SWS_RDS_90288]).
- Set num_lines field to value provided by the application. If value is not available or invalid, set this field to 0.
- Set i_seq_num field to value provided by the application. If value is not available or invalid, set this field to 0.
- Set line_number field to value provided by the application. If value is not available or invalid, set this field to 0.

⌋

Note for [SWS_RDS_10569]: Refer to Chapter 7.2.4.2.1.2 for details on AVTPDU-common-header fields and to Chapter 7.2.4.2.1.3 for details on AVTPDU-common-stream-header fields.

7.2.4.3 Error handling for consuming or producing IEEE1722 stream data

[SWS_RDS_10570] Error handling for read and write data to an disconnected `IEEE1722RawDataStream`

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If ReadData operation of am `IEEE1722RawDataStreamConsumer` instance or the WriteData operation of an `IEEE1722RawDataStreamProducer` instance is invoked and the communication connection of the corresponding data link layer socket is disconnected, then the operation shall do nothing and return with kStreamAlreadyDisconnected.⌋

[SWS_RDS_10571] Error handling for read and write data processing with system interrupt

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈If ReadData operation of am `IEEE1722RawDataStreamConsumer` instance or the WriteData operation of an `IEEE1722RawDataStreamProducer` instance is processed and the operation is interrupted by the system, then the operation shall reset all internal states, release internal resources and return with kInterruptedBySignal.⌋

7.3 Functional cluster lifecycle

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 [3] by [SWS_CORE_15005] and [SWS_CORE_90022].

7.3.1 Startup

No special startup handling is needed for the Raw Data Stream functional cluster (e.g. no state is maintained across power cycles).

7.3.2 Shutdown

No special shutdown handling is needed for the Raw Data Stream functional cluster.

7.4 Reporting

7.4.1 Security Events

This functional cluster does not define any security events.

7.4.2 Log Messages

This functional cluster does not define any non-verbose log messages (i.e., modelled DLT messages).

7.4.3 Violation Messages

[SWS_CORE_13003]

Dlt-Message	InstanceSpecifierMappingIntegrityViolation		
Description	InstanceSpecifier either cannot be resolved in the model in the context of your executable, or it refers to a model element other than a PortPrototype.		
MessageId	0x80001ffc		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	

▽

△

location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13004]

Dlt-Message	PortInterfaceMappingViolation		
Description	The type of mapping does not match the expected type of PortInterface: {portInterfaceTypeName} referenced by a {mappingTypeName}.		
MessageId	0x80001ffb		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13005]

Dlt-Message	ProcessMappingViolation		
Description	Matching InstanceRef exists, but no matching (modelled) Process found that matches the (runtime) process.		
MessageId	0x80001ffa		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

[SWS_CORE_13006]

Dlt-Message	InstanceSpecifierAlreadyInUseViolation
Description	Violation message that is sent in case a constructor in the ara framework was called with an Instance Specifier already in use in this process.

▽

△

MessageId	0x80001ff9		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator.	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example {filename}:{linenumber}.	uint8 [encoding UTF-8]	
instanceSpecifier	InstanceSpecifier used to try to create the object.	uint8 [encoding UTF-8]	
className	Name of the class that was instantiated.	uint8 [encoding UTF-8]	

7.4.4 Production Errrors

This functional cluster does not define any production errors (i.e., Diagnostic Events).

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)	
Port Interfaces:	States that the C++ API `class` is the related C++ API binding for the given modeled sub-class of `PortInterface`	
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 C++ `namespace` (in case of a `class` or non-member function) or a `class` declaration (in case of a member)	
Symbol:	C++ symbol 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 parameterize 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 C++ `Exception`s that may be thrown by the function	
Violations:	List of violations that may raised by the function	
Errors:	Error type (0..*)	List of defined `ara::core::ErrorCode`s 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 standardized by AUTOSAR
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

8.1 PortInterface to API class binding

This table shows the `API class` binding for each `PortInterface` owned by this functional cluster and those functions taking an `ara::core::InstanceSpecifier` argument, designated to "construct" that class. These constructing functions may be any combination of special-member constructors, named constructor members or non-member factory constructors.

Port Interface	API Class / Function
IEEE1722RawDataStreamConsumerInterface	[SWS_RDS_90311] Definition of API class ara::rds::IEEE1722RawDataStreamConsumer
	[SWS_RDS_90312] Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::Create
IEEE1722RawDataStreamProducerInterface	[SWS_RDS_90322] Definition of API class ara::rds::IEEE1722RawDataStreamProducer
	[SWS_RDS_90323] Definition of API function ara::rds::IEEE1722RawDataStreamProducer::Create
RawDataStreamClientInterface	[SWS_RDS_10481] Definition of API class ara::rds::RawDataStreamClient
	[SWS_RDS_10482] Definition of API function ara::rds::RawDataStreamClient::Create
RawDataStreamServerInterface	[SWS_RDS_11311] Definition of API class ara::rds::RawDataStreamServer
	[SWS_RDS_11312] Definition of API function ara::rds::RawDataStreamServer::Create

Table 8.2: PortInterface (sub-class) to API class / function binding

8.2 Header: ara/rds/raw_data_stream.h

8.2.1 Class: RawDataStreamClient

[SWS_RDS_10481] Definition of API class ara::rds::RawDataStreamClient

Upstream requirements: RS_CM_00410, RS_CM_00411

⌈

Kind:	class
Port Interfaces:	RawDataStreamClientInterface
Header file:	#include "ara/rds/raw_data_stream.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	RawDataStreamClient
Syntax:	`class RawDataStreamClient final {...};`
Description:	This class defines a RawDataStreamClient object for reading and writing binary data streams over a network connection. .

⌋

8.2.1.1 Public Member Functions

8.2.1.1.1 Special Member Functions

8.2.1.1.1.1 Copy Constructor

[SWS_RDS_11303] Definition of API function ara::rds::RawDataStreamClient::RawDataStreamClient

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamClient`
Syntax:	`RawDataStreamClient (const RawDataStreamClient &)=delete;`
Description:	Copy constructor of the RawDataStreamClient - not allowed.

⌋

8.2.1.1.1.2 Move Constructor

[SWS_RDS_11305] Definition of API function ara::rds::RawDataStreamClient::RawDataStreamClient

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00147

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`RawDataStreamClient (RawDataStreamClient &&other) noexcept;`	
Parameters (in):	other	The RawDataStreamClient object to be moved.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move constructor of the RawDataStreamClient.	

⌋

8.2.1.1.1.3 Copy Assignment Operator

[SWS_RDS_11304] Definition of API function ara::rds::RawDataStream Client::operator=

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00147

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamClient`
Syntax:	`RawDataStreamClient & operator= (const RawDataStreamClient &)=delete;`
Description:	Copy assignment operator of the RawDataStreamClient - not allowed.

⌋

8.2.1.1.1.4 Move Assignment Operator

[SWS_RDS_11306] Definition of API function ara::rds::RawDataStream Client::operator=

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00147

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`RawDataStreamClient & operator= (RawDataStreamClient &&other) & noexcept;`	
Parameters (in):	other	The RawDataStreamClient object to be moved.
Return value:	RawDataStreamClient &	The moved RawDataStreamClient object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of the RawDataStreamClient.	

⌋

8.2.1.1.1.5 Destructor

[SWS_RDS_10483] Definition of API function ara::rds::RawDataStream Client::~RawDataStreamClient

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamClient`
Syntax:	`~RawDataStreamClient () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the RawDataStreamClient that deletes the RawDataStreamClient instance. If the connection is still open, the connection is closed and shut down (calling Shutdown()) before destroying the RawDataStreamClient object.

⌋

8.2.1.1.2 Member Functions

8.2.1.1.2.1 Connect()

[SWS_RDS_10484] Definition of API function ara::rds::RawDataStream Client::Connect

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`ara::core::Result< void > Connect () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kConnection Refused	rollback_semantics
		The target address was not listening for connections or refused the connection request.
	RdsErrc::kAddressNot Available	rollback_semantics
		The specified address is not available from the local machine.
	RdsErrc::kStream AlreadyConnected	rollback_semantics
		The specified connection is already connected.
	RdsErrc::kPeer Unreachable	rollback_semantics
		Network error. The peer is unreachable (POSIX ENETUNREACH).

▽

△

	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Sets up a unicast socket connection for the RawDataStream defined by the instance, and establishes a connection to the TCP server. In the case of UDP, no connection is established.	

⌋

8.2.1.1.2.2 Connect(std::chrono::milliseconds)

[SWS_RDS_11307] Definition of API function ara::rds::RawDataStream Client::Connect

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`ara::core::Result< void > Connect (std::chrono::milliseconds timeout) noexcept;`	
Parameters (in):	timeout	Timeout value for this operation.
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kConnection Refused	rollback_semantics
		The target address was not listening for connections or refused the connection request.
	RdsErrc::kAddressNot Available	rollback_semantics
		The specified address is not available from the local machine.
	RdsErrc::k CommunicationTimeout	rollback_semantics
		The operation was not successful and timed out.
	RdsErrc::kStream AlreadyConnected	rollback_semantics
		The specified connection is already connected.
	RdsErrc::kPeer Unreachable	rollback_semantics
		Network error. The peer is unreachable (POSIX ENETUNREACH).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Sets up a unicast socket connection for the RawDataStream defined by the instance, and establishes a connection to the TCP server within a provided timeout value. In the case of UDP, no connection is established.	

⌋

8.2.1.1.2.3 ReadData(std::size_t, std::chrono::milliseconds)

[SWS_RDS_11309] Definition of API function ara::rds::RawDataStream Client::ReadData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`ara::core::Result< ReadDataResult > ReadData (std::size_t maxLength, std::chrono::milliseconds timeout) noexcept;`	
Parameters (in):	maxLength	The requested maximum number of bytes to read from the stream.
	timeout	Timeout value for this operation.
Return value:	ara::core::Result< ReadDataResult >	a struct of type ReadDataResult if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNotConnected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kCommunicationTimeout	rollback_semantics
		The operation was not successful and timed out.
	RdsErrc::kInterruptedBySignal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to read a number of bytes of data from the socket connection for the RawDataStream defined by the instance. A timeout value is provided for non-blocking operation.	

⌋

8.2.1.1.2.4 ReadData(std::size_t)

[SWS_RDS_10486] Definition of API function ara::rds::RawDataStream Client::ReadData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`ara::core::Result< ReadDataResult > ReadData (std::size_t maxLength) noexcept;`	
Parameters (in):	maxLength	The requested maximum number of bytes to read from the stream.
Return value:	ara::core::Result< ReadDataResult >	a struct of type ReadDataResult if succesful, otherwise an error code indicating the error.

▽

△

Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNotConnected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBySignal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to read a number of bytes of data from the socket connection for the RawDataStream defined by the instance.	

⌋

8.2.1.1.2.5 Shutdown

[SWS_RDS_10485] Definition of API function ara::rds::RawDataStreamClient::Shutdown

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`ara::core::Result< void > Shutdown () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNotConnected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBySignal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Closes the socket connection for the RawDataStream defined by the instance. Both the receiving and the sending part of the socket connection is shut down.	

⌋

8.2.1.1.2.6 WriteData(std::unique_ptr<ara::core::Byte[]>, std::size_t)

[SWS_RDS_10487] Definition of API function ara::rds::RawDataStream Client::WriteData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`ara::core::Result< std::size_t > WriteData (std::unique_ptr< ara::core::Byte[]> data, std::size_t maxLength) noexcept;`	
Parameters (in):	data	std::unique pointer to the byte array to send.
	maxLength	The requested maximum number of bytes to write to the stream.
Return value:	ara::core::Result< std::size_t >	the actual number of bytes written if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kConnection ClosedByPeer	rollback_semantics
		Network error. The established connection has been shut down during writing (POSIX EPIPE).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to write of a number of bytes to the the socket connection for the RawDataStream defined by the instance (for 1:1 use cases).	

⌋

8.2.1.1.2.7 WriteData(std::unique_ptr<ara::core::Byte[]>, std::size_t, std::chrono::milliseconds)

[SWS_RDS_11310] Definition of API function ara::rds::RawDataStream Client::WriteData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamClient`
Syntax:	`ara::core::Result< std::size_t > WriteData (std::unique_ptr< ara::core::Byte[]> data, std::size_t maxLength, std::chrono::milliseconds timeout) noexcept;`

▽

△

Parameters (in):	data	std::unique pointer to the byte array to send.
	maxLength	The requested maximum number of bytes to write to the stream.
	timeout	Timeout value for this operation.
Return value:	ara::core::Result< std::size_t >	the actual number of bytes written if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::k CommunicationTimeout	rollback_semantics
		The operation was not successful and timed out.
	RdsErrc::kConnection ClosedByPeer	rollback_semantics
		Network error. The established connection has been shut down during writing (POSIX EPIPE).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to write a number of bytes to the the socket connection for the RawDataStream defined by the instance. A timeout value is provided for non-blocking operation.	

⌋

8.2.1.1.3 Named Constructors

8.2.1.1.3.1 Create

[SWS_RDS_10482] Definition of API function ara::rds::RawDataStream Client::Create

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00170, RS_AP_00171, RS_AP_00172, RS_AP_00173

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamClient`	
Syntax:	`static ara::core::Result< RawDataStreamClient > Create (const ara::core::InstanceSpecifier &instance) noexcept;`	
Parameters (in):	instance	The instance specifier for the instance.
Return value:	ara::core::Result< Raw DataStreamClient >	ara::core::Result<RawDataStreamClient> The RawDataStream Client object if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	RdsErrc::kConnection CreationFailed	rollback_semantics
		Permission to create a connection is denied. (POSIX EACCES)

▽

△

	RdsErrc::kAddressNot Available	rollback_semantics
		The specified address is not available from the local machine.
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of the executable, or it refers to a model element other than a PortPrototype.
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `RawDataStreamMapping` needs to be typed by a `RawDataStreamClientInterface`
	`ProcessMappingViolation`	The type of mapping does not match the expected type of Port Interface
	`InstanceSpecifierAlreadyInUseViolation`	The constructor was called with an Instance Specifier already in use in this process
Description:	Named exception-less constructor that takes an instance Specifier qualifying the wanted network binding and parameters for the instance.	

⌋

8.2.2 Class: RawDataStreamServer

[SWS_RDS_11311] Definition of API class ara::rds::RawDataStreamServer

Upstream requirements: RS_CM_00410, RS_CM_00411

⌈

Kind:	class
Port Interfaces:	`RawDataStreamServerInterface`
Header file:	#include "ara/rds/raw_data_stream.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	RawDataStreamServer
Syntax:	`class RawDataStreamServer final {...};`
Description:	This class defines a RawDataStreamServer object for reading and writing binary data streams over a network connection. .

⌋

8.2.2.1 Public Member Functions

8.2.2.1.1 Special Member Functions

8.2.2.1.1.1 Move Constructor

[SWS_RDS_11316] Definition of API function ara::rds::RawDataStreamServer::RawDataStreamServer

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00147

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`RawDataStreamServer (RawDataStreamServer &&other) noexcept;`	
Parameters (in):	other	The RawDataStreamServer object to be moved.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move constructor of the RawDataStreamServer.	

⌋

8.2.2.1.1.2 Copy Constructor

[SWS_RDS_11314] Definition of API function ara::rds::RawDataStreamServer::RawDataStreamServer

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamServer`
Syntax:	`RawDataStreamServer (const RawDataStreamServer &)=delete;`
Description:	Copy constructor of the RawDataStreamServer - not allowed.

⌋

8.2.2.1.1.3 Move Assignment Operator

[SWS_RDS_11317] Definition of API function ara::rds::RawDataStream Server::operator=

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_-AP_00147

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`RawDataStreamServer & operator= (RawDataStreamServer &&other) & noexcept;`	
Parameters (in):	other	The RawDataStreamServer object to be moved.
Return value:	RawDataStreamServer &	The moved RawDataStreamServer object
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator of the RawDataStreamServer.	

⌋

8.2.2.1.1.4 Copy Assignment Operator

[SWS_RDS_11315] Definition of API function ara::rds::RawDataStream Server::operator=

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamServer`
Syntax:	`RawDataStreamServer & operator= (const RawDataStreamServer &)=delete;`
Description:	Copy assignment operator of the RawDataStreamServer - not allowed.

⌋

8.2.2.1.1.5 Destructor

[SWS_RDS_11313] Definition of API function ara::rds::RawDataStream Server::~RawDataStreamServer

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamServer`
Syntax:	`~RawDataStreamServer () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the RawDataStreamServer that deletes the RawDataStreamServer instance. If the connection is still open, the connection is closed and shut down (calling Shutdown()) before destroying the RawDataStreamClient object.

⌋

8.2.2.1.2 Member Functions

8.2.2.1.2.1 ReadData(std::size_t, std::chrono::milliseconds)

[SWS_RDS_11323] Definition of API function ara::rds::RawDataStream Server::ReadData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`ara::core::Result< ReadDataResult > ReadData (std::size_t maxLength, std::chrono::milliseconds timeout) noexcept;`	
Parameters (in):	maxLength	The requested maximum number of bytes to read from the stream.
	timeout	Parameter to assign a timeout for this operation.
Return value:	ara::core::Result< ReadDataResult >	a struct of type ReadDataResult.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNotConnected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kCommunicationTimeout	rollback_semantics
		The operation was not successful and timed out.
	RdsErrc::kInterruptedBySignal	no_rollback_semantics

▽

△

		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to read a number of bytes of data from the unicast socket connection for the RawData Stream defined by the instance. A timeout value is provided for non-blocking operation.	

⌋

8.2.2.1.2.2 ReadData(std::size_t)

[SWS_RDS_11322] Definition of API function ara::rds::RawDataStream Server::ReadData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`ara::core::Result< ReadDataResult > ReadData (std::size_t maxLength) noexcept;`	
Parameters (in):	maxLength	The requested maximum number of bytes to read from the stream.
Return value:	ara::core::Result< Read DataResult >	a struct of type ReadDataResult if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to read a number of bytes of data from the Unicast socket connection for the RawData Stream defined by the instance.	

⌋

8.2.2.1.2.3 Shutdown

[SWS_RDS_11320] Definition of API function ara::rds::RawDataStream Server::Shutdown

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`ara::core::Result< void > Shutdown () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Closes the socket connection for the RawDataStream defined by the instance. Both the receiving and the sending part of the socket connection is shut down.	

⌋

8.2.2.1.2.4 WaitForConnection(std::chrono::milliseconds)

[SWS_RDS_11319] Definition of API function ara::rds::RawDataStream Server::WaitForConnection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`ara::core::Result< void > WaitForConnection (std::chrono::milliseconds timeout) noexcept;`	
Parameters (in):	timeout	Timeout value for this operation.
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::k CommunicationTimeout	rollback_semantics
		The operation was not successful and timed out.
	RdsErrc::kConnection Aborted	rollback_semantics

▽

△

		Network error. The incoming connection was aborted (POSIX ECONNABORTED).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Enables the RawDataStreamServer instance for incoming TCP connections. For UDP connections the operation returns with no action. A timeout value is provided for non-blocking operation.	

⌋

8.2.2.1.2.5 WaitForConnection()

[SWS_RDS_11318] Definition of API function ara::rds::RawDataStream Server::WaitForConnection

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`ara::core::Result< void > WaitForConnection () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kConnection Aborted	rollback_semantics
		Network error. The incoming connection was aborted (POSIX ECONNABORTED).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Enables the RawDataStreamServer instance for incoming TCP connections. For UDP connections the operation returns with no action.	

⌋

8.2.2.1.2.6 WriteData(std::unique_ptr<ara::core::Byte[]>, std::size_t)

[SWS_RDS_11324] Definition of API function ara::rds::RawDataStream Server::WriteData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`ara::core::Result< std::size_t > WriteData (std::unique_ptr< ara::core::Byte[]> data, std::size_t maxLength) noexcept;`	
Parameters (in):	data	std::unique pointer to the byte array to send.
	maxLength	The requested maximum number of bytes to write to the stream.
Return value:	ara::core::Result< std::size_t >	the actual number of bytes written if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kConnection ClosedByPeer	rollback_semantics
		Network error. The established connection has been shut down during writing (POSIX EPIPE).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to write a number of bytes to the the socket connection for the RawDataStream defined by the instance.	

⌋

8.2.2.1.2.7 WriteData(std::unique_ptr<ara::core::Byte[]>, std::size_t, std::chrono::milliseconds)

[SWS_RDS_11325] Definition of API function ara::rds::RawDataStream Server::WriteData

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`class ara::rds::RawDataStreamServer`
Syntax:	`ara::core::Result< std::size_t > WriteData (std::unique_ptr< ara::core::Byte[]> data, std::size_t maxLength, std::chrono::milliseconds timeout) noexcept;`

▽

△

Parameters (in):	data	std::unique pointer to the byte array to send.
	maxLength	The requested maximum number of bytes to write to the stream.
	timeout	Parameter to assign a timeout for this operation.
Return value:	ara::core::Result< std::size_t >	the actual number of bytes written if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::k CommunicationTimeout	rollback_semantics
		The operation was not successful and timed out.
	RdsErrc::kConnection ClosedByPeer	rollback_semantics
		Network error. The established connection has been shut down during writing (POSIX EPIPE).
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Requests to write a number of bytes to the the socket connection for the RawDataStream defined by the instance. A timeout value is provided for non-blocking operation.	

⌋

8.2.2.1.3 Named Constructors

8.2.2.1.3.1 Create

[SWS_RDS_11312] Definition of API function ara::rds::RawDataStream Server::Create

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412, RS_AP_00145, RS_AP_00170, RS_AP_00171, RS_AP_00172, RS_AP_00173

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream.h"	
Scope:	`class ara::rds::RawDataStreamServer`	
Syntax:	`static ara::core::Result< RawDataStreamServer > Create (const ara::core::InstanceSpecifier &instance) noexcept;`	
Parameters (in):	instance	The instance specifier for the instance.
Return value:	ara::core::Result< Raw DataStreamServer >	ara::core::Result<RawDataStreamServer> The RawDataStream Server object if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	RdsErrc::kConnection CreationFailed	rollback_semantics
		Permission to create a connection is denied. (POSIX EACCES)

▽

△

	RdsErrc::kAddressNotAvailable	rollback_semantics
		The specified address is not available from the local machine.
	RdsErrc::kStreamAlreadyConnected	rollback_semantics
		The specified connection is already connected.
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of the executable, or it refers to a model element other than a PortPrototype.
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `RawDataStreamMapping` needs to be typed by a `RawDataStreamServerInterface`
	`ProcessMappingViolation`	The type of mapping does not match the expected type of Port Interface
	`InstanceSpecifierAlreadyInUseViolation`	The constructor was called with an Instance Specifier already in use in this process
Description:	Named exception-less constructor that takes an instance Specifier qualifying the wanted network credentials (UDP or TCP) for the instance. A socket is created and bound to the address and port specified in the local credentials.	

⌋

8.2.3 Struct: ReadDataResult

[SWS_RDS_11300] Definition of API class ara::rds::ReadDataResult

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	struct
Header file:	#include "ara/rds/raw_data_stream.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	ReadDataResult
Syntax:	`struct ReadDataResult {...};`
Description:	The ReadDataResult struct used as return value from ReadData().

⌋

8.2.3.1 Public Member Variables

8.2.3.1.1 data

[SWS_RDS_11301] Definition of API variable ara::rds::ReadDataResult::data

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`struct ara::rds::ReadDataResult`
Symbol:	data
Type:	std::unique_ptr< ara::core::Byte[]>
Syntax:	`std::unique_ptr<ara::core::Byte[]> data;`
Description:	std::unique pointer to the read data.

⌋

8.2.3.1.2 numberOfBytes

[SWS_RDS_11302] Definition of API variable ara::rds::ReadDataResult::numberOfBytes

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream.h"
Scope:	`struct ara::rds::ReadDataResult`
Symbol:	numberOfBytes
Type:	std::size_t
Syntax:	`std::size_t numberOfBytes;`
Description:	The actual number of bytes read from the stream.

⌋

8.3 Header: ara/rds/raw_data_stream_IEEE1722.h

8.3.1 Class: IEEE1722Datagram

[SWS_RDS_90225] Definition of API class ara::rds::IEEE1722Datagram

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722Datagram
Syntax:	`class IEEE1722Datagram {...};`
Description:	This class defines the base class which represents IEEE1722 payload information that is interchanged between an application and the ARA::RDS functional cluster.

⌋

8.3.1.1 Protected Member Variables

8.3.1.1.1 data

[SWS_RDS_90226] Definition of API variable ara::rds::IEEE1722Datagram::data

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram`
Symbol:	data
Type:	std::unique_ptr< ara::core::Byte[]>
Syntax:	`std::unique_ptr<ara::core::Byte[]> data;`
Description:	std::unique pointer member variable which represents the payload of an IEEE1722 stream. For reception, data represents a pointer to the data received as payload via an IEEE1722 stream For transmission, data represents a pointer to the data which is transmitted as payload via an IEEE1722 stream
Visibility:	protected

⌋

8.3.1.1.2 stream_data_length

[SWS_RDS_90227] Definition of API variable ara::rds::IEEE1722Datagram::stream_data_length

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram`
Symbol:	stream_data_length
Type:	std::uint16_t
Syntax:	`std::uint16_t stream_data_length;`
Description:	std::uint16_t member variable which represents the IEEE1722 defined stream_data_length header field of an IEEE1722 stream The value range for stream_data_length shall be: 0x00 00 ... 0xFF FF: valid For reception, stream_data_length contain the number of data bytes received as payload via an IEEE1722 stream For transmission, stream_data_length contain the number of data bytes transmitted as payload via an IEEE1722 stream
Visibility:	protected

⌋

8.3.1.2 Public Member Functions

8.3.1.2.1 Special Member Functions

8.3.1.2.1.1 Destructor

[SWS_RDS_90228] Definition of API function ara::rds::IEEE1722Datagram::~IEEE1722Datagram

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram`
Syntax:	`~IEEE1722Datagram () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722Datagram that deletes the IEEE1722Datagram instance.

⌋

8.3.2 Class: IEEE1722Datagram61883_IIDC

[SWS_RDS_90243] Definition of API class ara::rds::IEEE1722Datagram61883_IIDC

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722Datagram61883_IIDC
Base class:	IEEE1722DatagramAVTPDUCommonStreamHeaderFields
Syntax:	`class IEEE1722Datagram61883_IIDC final : protected IEEE1722Datagram AVTPDUCommonStreamHeaderFields {...};`
Description:	This class defines an object which represents the IEEE1722 definded subtype 61883_IIDC header fields. Data length (stream_data_length) and payload (data) is derived from class IEEE1722Datagram.

⌋

8.3.2.1 Protected Member Variables

8.3.2.1.1 channel

[SWS_RDS_90245] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::channel

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	channel
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> channel;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined channel 6 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for channel shall be: 0x00...0x3F : valid 0x40...0xFF : not used For transmission, the value of channel shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.2 dbc

[SWS_RDS_90254] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::dbc

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	dbc
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> dbc;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined dbc (data block count) 8 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for dbc shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, the value of dbc shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.3 dbs

[SWS_RDS_90250] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::dbs

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	dbs
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> dbs;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined dbs (data block size) 8 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for dbs shall be: 0x00 00...0x00 FF : valid 0x00 01...0xFF FF : not used For transmission, the value of dbs shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.4 fdf_with_sph

[SWS_RDS_90259] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fdf_with_sph

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	fdf_with_sph
Type:	ara::core::Optional< std::uint32_t >
Syntax:	`ara::core::Optional<std::uint32_t> fdf_with_sph;`
Description:	std::uint32_t optional member variable which represents the IEEE1722 defined fdf_with_sph (format dependent field) 24 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value shall be considered, if the sph header field value is set to 1. The value range for fdf_with_sph shall be: 0x00 00 00 00 ... 0x00 FF FF FF : valid 0x01 FF FF FF ... 0xFF FF FF FF : not used For transmission, the value of fdf_with_sph shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.5 fdf_without_sph

[SWS_RDS_90257] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fdf_without_sph

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	fdf_without_sph
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> fdf_without_sph;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined fdf_without_sph (format dependent field) 8 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for fdf_without_sph shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, the value of fdf_without_sph shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.6 fmt

[SWS_RDS_90256] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fmt

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	fmt
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> fmt;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined fmt (stream format) 6 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for fmt shall be: 0x00...0x3F : valid 0x4F...0xFF : not used For transmission, the value of fmt shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.7 fn

[SWS_RDS_90251] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fn

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	fn
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> fn;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined fn (fraction number) 2 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for fn shall be: 0x00...0x03 : valid 0x04...0xFF : not used For transmission, the value of fn shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.8 qi_1

[SWS_RDS_90248] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::qi_1

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	qi_1
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> qi_1;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined qi_1 (quadlet indicator) 2 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for qi_1 shall be: 0x00...0x03 : valid 0x04...0xFF : not used For transmission, the value of qi_1 shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.9 qi_2

[SWS_RDS_90255] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::qi_2

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	qi_2
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> qi_2;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined qi_2 (quadlet indicator) 2 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for qi_2 shall be: 0x00...0x03 : valid 0x04...0xFF : not used For transmission, the value of qi_2 shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.10 qpc

[SWS_RDS_90252] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::qpc

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	qpc
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> qpc;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined qpc (quadlet padding count) 3 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for qpc shall be: 0x00...0x07 : valid 0x08...0xFF : not used For transmission, the value of qpc shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.11 sid

[SWS_RDS_90249] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::sid

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	sid
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> sid;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined sid (source indentifier) 6 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for sid shall be: 0x00...0x3F : valid 0x40...0xFF : not used For transmission, the value of sid shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.12 sph

[SWS_RDS_90253] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::sph

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	sph
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> sph;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined sph (source package header) 1 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for sph shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the value of sph shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.13 sy

[SWS_RDS_90247] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::sy

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	sy
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> sy;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined sy 4 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for sy shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the value of sy shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.14 syt

[SWS_RDS_90258] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::syt

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	syt
Type:	ara::core::Optional< std::uint32_t >
Syntax:	`ara::core::Optional<std::uint32_t> syt;`
Description:	std::uint32_t optional member variable which represents the IEEE1722 defined syt (synchronization timing) 16 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for syt shall be: 0x00 00 00 00 ... 0x00 00 FF FF : valid 0x00 01 00 00 ... 0xFF FF FF FF : not used For transmission, the value of syt shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.15 tag

[SWS_RDS_90244] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::tag

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	tag
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> tag;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined tag 2 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for tag shall be: 0x00...0x03 : valid 0x04...0xFF : not used For transmission, the value of tag shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.1.16 tcode

[SWS_RDS_90246] Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::tcode

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Symbol:	tcode
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> tcode;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined tcode (type code) 4 bit header field of an IEEE1722 stream with subtype IEC61883_IIDC. The value range for tcode shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the value of tcode shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.2.2 Public Member Functions

8.3.2.2.1 Special Member Functions

8.3.2.2.1.1 Destructor

[SWS_RDS_90260] Definition of API function ara::rds::IEEE1722Datagram61883_IIDC::~IEEE1722Datagram61883_IIDC

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722Datagram61883_IIDC`
Syntax:	`~IEEE1722Datagram61883_IIDC () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722Datagram61883_IIDC that deletes the IEEE1722Datagram61883_IIDC instance.

⌋

8.3.3 Class: IEEE1722DatagramAAF

[SWS_RDS_90261] Definition of API class ara::rds::IEEE1722DatagramAAF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramAAF
Base class:	IEEE1722DatagramAVTPDUCommonStreamHeaderFields
Syntax:	`class IEEE1722DatagramAAF final : protected IEEE1722Datagram AVTPDUCommonStreamHeaderFields {...};`
Description:	This class defines an object which represents the IEEE1722 definded subtype AAF header fields. Data length (stream_data_length) and payload (data) is derived from class IEEE1722 Datagram.

⌋

8.3.3.1 Protected Member Variables

8.3.3.1.1 aes3_data_type_h

[SWS_RDS_90270] Definition of API variable ara::rds::IEEE1722DatagramAAF::aes3_data_type_h

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	aes3_data_type_h
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> aes3_data_type_h;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined aes3_data_type_h 8 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, if format is set to a value which represents a AES3 stream data encapsulation the middleware shall set value of aes3_data_type_h to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.2 aes3_data_type_l

[SWS_RDS_90272] Definition of API variable ara::rds::IEEE1722Datagram AAF::aes3_data_type_l

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	aes3_data_type_l
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> aes3_data_type_l;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined aes3_data_type_l 8 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, if format is set to a value which represents a AES3 stream data encapsulation the middleware shall set value of aes3_data_type_h to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.3 aes3_dt_ref

[SWS_RDS_90271] Definition of API variable ara::rds::IEEE1722Datagram AAF::aes3_dt_ref

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	aes3_dt_ref
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> aes3_dt_ref;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined aes3_dt_ref (aes3_data_type reference) 3 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00...0x07 : valid 0x08...0xFF : not used For transmission, if format is set to a value which represents a AES3 stream data encapsulation the middleware shall set value of aes3_data_type_h to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.4 bit_depth

[SWS_RDS_90267] Definition of API variable ara::rds::IEEE1722DatagramAAF::bit_depth

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	bit_depth
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> bit_depth;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined bit_depth 8 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, the middleware shall set value of bit_depth to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.5 channels_per_frame

[SWS_RDS_90266] Definition of API variable ara::rds::IEEE1722DatagramAAF::channels_per_frame

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	channels_per_frame
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> channels_per_frame;`
Description:	std::uint16_t member variable which represents the IEEE1722 defined channels_per_frame 10 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00 00...0x03 FF : valid 0x00 40...0xFF FF : not used For transmission, the middleware shall set value of channels_per_frame to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.6 evt

[SWS_RDS_90264] Definition of API variable ara::rds::IEEE1722DatagramAAF::evt

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	evt
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> evt;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined event 4 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the middleware shall set value of evt to 0, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.7 format

[SWS_RDS_90262] Definition of API variable ara::rds::IEEE1722DatagramAAF::format

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	format
Type:	std::uint8_t
Syntax:	`std::uint8_t format;`
Description:	std::uint8_t member variable which represents the IEEE1722 defined format 8 bit header field of an IEEE1722 stream with subtype AAF. The following format values are defined: 0x00 (User; represents PCM stream data encapsulation) 0x01 (FLOAT_32BIT;represents PCM stream data encapsulation) 0x02 (INT_32BIT; represents PCM stream data encapsulation) 0x03 (INT_24BIT; represents PCM stream data encapsulation) 0x04 (INT_16BIT; represents PCM stream data encapsulation) 0x05 (AES3_32BIT: represents AES3 stream data encapsulation) 0x06 ... 0xFF : reserverd For transmission, the value shall be provided by the application.
Visibility:	protected

⌋

8.3.3.1.8 nfr

[SWS_RDS_90268] Definition of API variable ara::rds::IEEE1722DatagramAAF::nfr

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	nfr
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> nfr;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined nfr (nominal frame rate) 4 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, if format is set to a value which represents a AES3 stream data encapsulation the middleware shall set value of bit_depth to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.9 nsr

[SWS_RDS_90265] Definition of API variable ara::rds::IEEE1722DatagramAAF::nsr

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	nsr
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> nsr;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined nsr (nominal sample rate) 4 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the middleware shall set value of nsr to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.3.1.10 sp

[SWS_RDS_90263] Definition of API variable ara::rds::IEEE1722DatagramAAF::sp

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	sp
Type:	std::uint8_t
Syntax:	`std::uint8_t sp;`
Description:	std::uint8_t member variable which represents the IEEE1722 defined sp (sparse timestamp) 1 bit header field of an IEEE1722 stream with subtype AAF. The value range for sparse timestamp shall be: 0x00...0x01 : valid 0x01...0xFF : not used For transmission, the value shall be provided by the application.
Visibility:	protected

⌋

8.3.3.1.11 streams_per_frame

[SWS_RDS_90269] Definition of API variable ara::rds::IEEE1722DatagramAAF::streams_per_frame

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Symbol:	streams_per_frame
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> streams_per_frame;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined streams_per_frame 10 bit header field of an IEEE1722 stream with subtype AAF. The value range for event shall be: 0x00 00...0x03 FF : valid 0x04 00...0xFF FF : not used For transmission, if format is set to a value which represents a AES3 stream data encapsulation, then the middleware shall set value of streams_per_frame to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.3.2 Public Member Functions

8.3.3.2.1 Special Member Functions

8.3.3.2.1.1 Destructor

[SWS_RDS_90273] Definition of API function ara::rds::IEEE1722DatagramAAF::~IEEE1722DatagramAAF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAAF`
Syntax:	`~IEEE1722DatagramAAF () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramAAF that deletes the IEEE1722DatagramAAF instance.

⌋

8.3.4 Class: IEEE1722DatagramAVTPDUCommonHeaderFields

[SWS_RDS_90229] Definition of API class ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramAVTPDUCommonHeaderFields
Base class:	IEEE1722Datagram
Syntax:	`class IEEE1722DatagramAVTPDUCommonHeaderFields : protected IEEE1722Datagram {...};`
Description:	This class defines a IEEE1722DatagramAVTPDUCommonHeaderFields object which represents IEEE1722 defined AVTPDU common header fields. Those header fields are shared between all IEEE1722 defined AVTP stream data subtypes.

⌋

8.3.4.1 Protected Member Variables

8.3.4.1.1 avtpStreamDataSubtype

[SWS_RDS_90230] Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields::avtpStreamDataSubtype

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields`
Symbol:	avtpStreamDataSubtype
Type:	std::uint16_t
Syntax:	`std::uint16_t avtpStreamDataSubtype;`
Description:	std::uint16_t member variable which represents the used IEEE1722 defined AVTP stream data subtype header field. The following AVTP stream data subtypes shall be supported: 0x00 (61883_IIDC) 0x02 (AAF) 0x04 (CRF) 0x05 (TSCF) 0x07 (RVF) 0x82 (NTSCF)
Visibility:	protected

⌋

8.3.4.1.2 headerSpecific

[SWS_RDS_90231] Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields::headerSpecific

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields`
Symbol:	headerSpecific
Type:	std::uint8_t
Syntax:	`std::uint8_t headerSpecific;`
Description:	std::uint8_t member variable which represents the IEEE1722 defined header specific 1 bit header field. The value range for headerSpecific shall be: 0x00...0x01 : valid 0x02...0xFF : not used
Visibility:	protected

⌋

8.3.4.1.3 version

[SWS_RDS_90232] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonHeaderFields::version

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields`
Symbol:	version
Type:	std::uint8_t
Syntax:	`std::uint8_t version;`
Description:	std::uint8_t member variable which represents the IEEE1722 defined version 3 bit header field. The value range for version shall be: 0x00...0x07 : valid 0x08...0xFF : not used
Visibility:	protected

⌋

8.3.4.2 Public Member Functions

8.3.4.2.1 Special Member Functions

8.3.4.2.1.1 Destructor

[SWS_RDS_90233] Definition of API function ara::rds::IEEE1722DatagramAVTP-DUCommonHeaderFields::~IEEE1722DatagramAVTPDUCommonHeaderFields

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonHeaderFields`
Syntax:	`~IEEE1722DatagramAVTPDUCommonHeaderFields () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramAVTPDUCommonHeaderFields that deletes the IEEE1722 DatagramAVTPDUCommonHeaderFields instance.

⌋

8.3.5 Class: IEEE1722DatagramAVTPDUCommonStreamHeaderFields

[SWS_RDS_90234] Definition of API class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramAVTPDUCommonStreamHeaderFields
Base class:	IEEE1722DatagramAVTPDUCommonHeaderFields
Syntax:	`class IEEE1722DatagramAVTPDUCommonStreamHeaderFields : protected IEEE1722DatagramAVTPDUCommonHeaderFields {...};`
Description:	This class defines a IEEE1722DatagramAVTPDUCommonStreamHeaderFields object which represents IEEE1722 defined AVTPDU common stream header fields. Those header fields are shared between the following IEEE1722 defined AVTP stream data subtypes: 61883_IIDC, AAF, RVF and TSCF.

⌋

8.3.5.1 Protected Member Variables

8.3.5.1.1 avtp_timestamp

[SWS_RDS_90237] Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields::avtp_timestamp

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	avtp_timestamp
Type:	ara::core::Optional< ara::tsync::Timestamp >
Syntax:	`ara::core::Optional<ara::tsync::Timestamp> avtp_timestamp;`
Description:	ara::tsync::Timestamp optional member variable which represents the IEEE1722 defined avtp_timestamp (a.k.a AVTP presentation time) 32 bit header field used for this IEEE1722 stream. The value range for avtp_timestamp shall be: 0x00 00 00 00 00 00 00 00 ... 0x00 00 00 00 FF FF FF FF: valid 0x00 00 00 01 00 00 00 00 ... 0xFF FF FF FF FF FF FF FF: not used For transmission, the middleware shall calculate the AVTP presentation time, if it is not provided by the application.
Visibility:	protected

⌋

8.3.5.1.2 mac_address

[SWS_RDS_90235] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonStreamHeaderFields::mac_address

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	mac_address
Type:	ara::core::Optional< std::uint64_t >
Syntax:	`ara::core::Optional<std::uint64_t> mac_address;`
Description:	std::uint64_t optional member variable which represents the MAC address part of the IEEE1722 defined stream id header field. This is used for a unambiguous identification of the IEEE1722 stream. Please note, the stream id consist of the following fragment: 48bit MAC address and 16bit unique id. The value range for mac_address shall be: 0x00 00 00 00 00 00 00 00 ... 0x00 00 FF FF FF FF FF FF: valid 0x00 01 00 00 00 00 00 00 ... 0xFF FF FF FF FF FF FF FF: not used For transmission, the middleware shall set the MAC address part which is configured, if it is not provided by the application.
Visibility:	protected

⌋

8.3.5.1.3 mr

[SWS_RDS_90238] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonStreamHeaderFields::mr

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	mr
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> mr;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined mr (media clock restart) 1 bit header field. The value range for version shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the middleware shall set value of mr to 0, if it is not provided by the application.
Visibility:	protected

⌋

8.3.5.1.4 sequenceNum

[SWS_RDS_90240] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonStreamHeaderFields::sequenceNum

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	sequenceNum
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> sequenceNum;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined sequence number 8 bit header field. The value range for version shall be: 0x00...0xFF : valid For transmission, the middleware shall increase the sequence number with 0x01 for each transmission request, if it is not provided by the application. If the sequence number reaches the maximum value, then it should re-start with 0x00.
Visibility:	protected

⌋

8.3.5.1.5 tu

[SWS_RDS_90241] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonStreamHeaderFields::tu

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	tu
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> tu;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined tu (avtp_timestamp uncertain) 1 bit header field. The value range for version shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the middleware shall set value of tu according the current state of the corresponding global time base, if it is not provided by the application.
Visibility:	protected

⌋

8.3.5.1.6 tv

[SWS_RDS_90239] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonStreamHeaderFields::tv

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	tv
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> tv;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined tv (avtp timestamp valid) 1 bit header field. The value range for version shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the middleware shall set value of tv to 0, if it is not provided by the application.
Visibility:	protected

⌋

8.3.5.1.7 unique_id

[SWS_RDS_90236] Definition of API variable ara::rds::IEEE1722DatagramAVTP-DUCommonStreamHeaderFields::unique_id

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Symbol:	unique_id
Type:	ara::core::Optional< std::uint64_t >
Syntax:	`ara::core::Optional<std::uint64_t> unique_id;`
Description:	std::uint16_t optional member variable which represents the unique id part of the IEEE1722 defined stream id header field. This is used for a unambiguous identification of the IEEE1722 stream. Please note, the stream id consist of the following fragment: 48bit MAC address and 16bit unique id. The value range for mac_address shall be: 0x00 00 ... 0xFF FF: valid For transmission, the middleware shall set the unique address part which is configured, if it is not provided by the application.
Visibility:	protected

⌋

8.3.5.2 Public Member Functions

8.3.5.2.1 Special Member Functions

8.3.5.2.1.1 Destructor

[SWS_RDS_90242] Definition of API function ara::rds::IEEE1722Datagram AVTPDUCommonStreamHeaderFields::~IEEE1722DatagramAVTPDUCommon StreamHeaderFields

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramAVTPDUCommonStreamHeaderFields`
Syntax:	`~IEEE1722DatagramAVTPDUCommonStreamHeaderFields () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramAVTPDUCommonStreamHeaderFields that deletes the IEEE1722DatagramAVTPDUCommonStreamHeaderFields instance.

⌋

8.3.6 Class: IEEE1722DatagramCRF

[SWS_RDS_90293] Definition of API class ara::rds::IEEE1722DatagramCRF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramCRF
Base class:	IEEE1722DatagramAVTPDUCommonHeaderFields
Syntax:	`class IEEE1722DatagramCRF final : protected IEEE1722Datagram AVTPDUCommonHeaderFields {...};`
Description:	This class defines an object which represents the IEEE1722 definded subtype CRF header fields Data length (stream_data_length) and payload (data) is derived from class IEEE1722Datagram. Data length and payload representing the IEEE1722 defined crf_data_length and crf_data.

⌋

8.3.6.1 Protected Member Variables

8.3.6.1.1 base_frequency

[SWS_RDS_90302] Definition of API variable ara::rds::IEEE1722DatagramCRF::base_frequency

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	base_frequency
Type:	ara::core::Optional< std::uint32_t >
Syntax:	`ara::core::Optional<std::uint32_t> base_frequency;`
Description:	std::uint32_t optinal member variable which represents the IEEE1722 defined base_frequency 29 bit header field of an IEEE1722 stream with subtype CRF. The value range for event shall be: 0x00 00 00 00 ... 0x1F FF FF FF : valid 0x20 00 00 00 ... 0xFF FF FF FF : not used For transmission, the middleware shall set the value of base_frequency to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.2 fs

[SWS_RDS_90295] Definition of API variable ara::rds::IEEE1722DatagramCRF::fs

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	fs
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> fs;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined fs (frame sync) 1 bit header field. The value range for version shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the middleware shall set value to 0, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.3 mac_address

[SWS_RDS_90299] Definition of API variable ara::rds::IEEE1722DatagramCRF::mac_address

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	mac_address
Type:	ara::core::Optional< std::uint64_t >
Syntax:	`ara::core::Optional<std::uint64_t> mac_address;`
Description:	std::uint64_t optinal member variable which represents the MAC address part of the IEEE1722 defined stream id header field. This is used for a unambiguous identification of the IEEE1722 stream. Please note, the stream id consist of the following fragment: 48bit MAC address and 16bit unique id. The value range for mac_address shall be: 0x00 00 00 00 00 00 00 00 ... 0x00 00 FF FF FF FF FF FF: valid 0x00 01 00 00 00 00 00 00 ... 0xFF FF FF FF FF FF FF FF: not used For transmission, the middleware shall set the MAC address part which is configured, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.4 mr

[SWS_RDS_90294] Definition of API variable ara::rds::IEEE1722DatagramCRF::mr

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	mr
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> mr;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined mr (media clock restart) 1 bit header field. The value range for version shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the middleware shall set value to 0, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.5 pull

[SWS_RDS_90301] Definition of API variable ara::rds::IEEE1722DatagramCRF::pull

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	pull
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> pull;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined pull 3 bit header field of an IEEE1722 stream with subtype CRF. The value range for event shall be: 0x00...0x07 : valid 0x08...0xFF : not used For transmission, the middleware shall set the value of pull to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.6 sequenceNum

[SWS_RDS_90297] Definition of API variable ara::rds::IEEE1722DatagramCRF::sequenceNum

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	sequenceNum
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> sequenceNum;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined sequence number header field. The value range for version shall be: 0x00...0xFF : valid For transmission, the middleware shall increase the sequence number with 0x01 for each transmission request, if it is not provided by the application. If the sequence number reaches the maximum value, then it should re-start with 0x00.
Visibility:	protected

⌋

8.3.6.1.7 timestamp_interval

[SWS_RDS_90303] Definition of API variable ara::rds::IEEE1722DatagramCRF::timestamp_interval

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	timestamp_interval
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> timestamp_interval;`
Description:	std::uint16_t optinal member timestamp_interval which represents the IEEE1722 defined timestamp_interval 16 bit header field of an IEEE1722 stream with subtype CRF. The value range for event shall be: 0x00 00 ... 0xFF FF : valid For transmission, the middleware shall set the value of timestamp_interval to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.8 tu

[SWS_RDS_90296] Definition of API variable ara::rds::IEEE1722DatagramCRF::tu

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	tu
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> tu;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined tu (avtp_timestamp uncertain) 1 bit header field. The value range for version shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the middleware shall set value according the current state of the corresponding global time base, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.9 type

[SWS_RDS_90298] Definition of API variable ara::rds::IEEE1722DatagramCRF::type

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	type
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> type;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined type 8 bit header field. The value range for type shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the middleware shall set value to 0x00, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.1.10 unique_id

[SWS_RDS_90300] Definition of API variable ara::rds::IEEE1722DatagramCRF::unique_id

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Symbol:	unique_id
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> unique_id;`
Description:	std::uint16_t optinal member variable which represents the unique id part of the IEEE1722 defined stream id header field. This is used for a unambiguous identification of the IEEE1722 stream. Please note, the stream id consist of the following fragment: 48bit MAC address and 16bit unique id. The value range for mac_address shall be: 0x00 00 ... 0xFF FF: valid For transmission, the middleware shall set the unique address part which is configured, if it is not provided by the application.
Visibility:	protected

⌋

8.3.6.2 Public Member Functions

8.3.6.2.1 Special Member Functions

8.3.6.2.1.1 Destructor

[SWS_RDS_90304] Definition of API function ara::rds::IEEE1722DatagramCRF::~IEEE1722DatagramCRF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramCRF`
Syntax:	`~IEEE1722DatagramCRF () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramCRF that deletes the IEEE1722DatagramCRF instance.

⌋

8.3.7 Class: IEEE1722DatagramNTSCF

[SWS_RDS_90305] Definition of API class ara::rds::IEEE1722DatagramNTSCF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramNTSCF
Base class:	IEEE1722DatagramAVTPDUCommonHeaderFields
Syntax:	`class IEEE1722DatagramNTSCF final : protected IEEE1722Datagram AVTPDUCommonHeaderFields {...};`
Description:	This class defines an object which represents the IEEE1722 definded subtype NTSCF header fields. Payload (data) is derived from class IEEE1722Datagram. Payload represents the IEEE1722 defined acf_payload_data. The data length is defined by the member variable ntscf_data_length.

⌋

8.3.7.1 Protected Member Variables

8.3.7.1.1 mac_address

[SWS_RDS_90308] Definition of API variable ara::rds::IEEE1722DatagramNTSCF::mac_address

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramNTSCF`
Symbol:	mac_address
Type:	ara::core::Optional< std::uint64_t >
Syntax:	`ara::core::Optional<std::uint64_t> mac_address;`
Description:	std::uint64_t optinal member variable which represents the MAC address part of the IEEE1722 defined stream id header field. This is used for a unambiguous identification of the IEEE1722 stream. Please note, the stream id consist of the following fragment: 48bit MAC address and 16bit unique id. The value range for mac_address shall be: 0x00 00 00 00 00 00 00 00 ... 0x00 00 FF FF FF FF FF FF: valid 0x00 01 00 00 00 00 00 00 ... 0xFF FF FF FF FF FF FF FF: not used For transmission, the middleware shall set the MAC address part which is configured, if it is not provided by the application.
Visibility:	protected

⌋

8.3.7.1.2 ntscf_data_length

[SWS_RDS_90306] Definition of API variable ara::rds::IEEE1722DatagramNTSCF::ntscf_data_length

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramNTSCF`
Symbol:	ntscf_data_length
Type:	std::uint16_t
Syntax:	`std::uint16_t ntscf_data_length;`

▽

△

Description:	std::uint16_t member variable which represents the IEEE1722 defined ntscf_data_length 11 bit header field of an IEEE1722 stream. This member variable replaces the stream_data_length from base class IEEE1722Datagram. shall not be set Base class IEEE1722Datagram shall not be set. The value range for ntscf_data_length shall be: 0x00 00 ... 0x03 FF: valid 0x04 00 ... 0xFF FF: not used For reception, ntscf_data_length contain the number of data bytes received as payload via an IEEE1722 stream For transmission, ntscf_data_length contain the number of data bytes transmitted as acf_payload_data via an IEEE1722 stream
Visibility:	protected

⌋

8.3.7.1.3 sequenceNum

[SWS_RDS_90307] Definition of API variable ara::rds::IEEE1722Datagram NTSCF::sequenceNum

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramNTSCF`
Symbol:	sequenceNum
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> sequenceNum;`
Description:	std::uint8_t optinal member variable which represents the IEEE1722 defined sequence number 8 bit header field. The value range for version shall be: 0x00...0xFF : valid For transmission, the middleware shall increase the sequence number with 0x01 for each transmission request, if it is not provided by the application. If the sequence number reaches the maximum value, then it should re-start with 0x00.
Visibility:	protected

⌋

8.3.7.1.4 unique_id

[SWS_RDS_90309] Definition of API variable ara::rds::IEEE1722DatagramNTSCF::unique_id

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramNTSCF`
Symbol:	unique_id
Type:	ara::core::Optional< std::uint64_t >
Syntax:	`ara::core::Optional<std::uint64_t> unique_id;`
Description:	std::uint16_t optinal member variable which represents the unique id part of the IEEE1722 defined stream id header field. This is used for a unambiguous identification of the IEEE1722 stream. Please note, the stream id consist of the following fragment: 48bit MAC address and 16bit unique id. The value range for mac_address shall be: 0x00 00 ... 0xFF FF: valid For transmission, the middleware shall set the unique address part which is configured, if it is not provided by the application.
Visibility:	protected

⌋

8.3.7.2 Public Member Functions

8.3.7.2.1 Special Member Functions

8.3.7.2.1.1 Destructor

[SWS_RDS_90310] Definition of API function ara::rds::IEEE1722DatagramNTSCF::~IEEE1722DatagramNTSCF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramNTSCF`
Syntax:	`~IEEE1722DatagramNTSCF () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramNTSCF that deletes the IEEE1722DatagramNTSCF instance.

⌋

8.3.8 Class: IEEE1722DatagramRVF

[SWS_RDS_90276] Definition of API class ara::rds::IEEE1722DatagramRVF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramRVF
Base class:	IEEE1722DatagramAVTPDUCommonStreamHeaderFields
Syntax:	`class IEEE1722DatagramRVF final : protected IEEE1722Datagram AVTPDUCommonStreamHeaderFields {...};`
Description:	This class defines an object which represents the IEEE1722 definded subtype RVF header fields. Data length (stream_data_length) and payload (data) is derived from class IEEE1722 Datagram.

⌋

8.3.8.1 Protected Member Variables

8.3.8.1.1 active_pixels

[SWS_RDS_90277] Definition of API variable ara::rds::IEEE1722DatagramRVF::active_pixels

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	active_pixels
Type:	ara::core::Optional< std::uint32_t >
Syntax:	`ara::core::Optional<std::uint32_t> active_pixels;`
Description:	std::uint32_t optional member variable which represents the IEEE1722 defined active_pixels 16 bit header field of an IEEE1722 stream with subtype RVF. The value range for active_pixels shall be: 0x00 00 00 00 ... 0x00 00 FF FF : valid 0x10 01 00 00 ... 0xFF FF FF FF : not used For transmission, the value of active_pixels shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.2 ap

[SWS_RDS_90279] Definition of API variable ara::rds::IEEE1722DatagramRVF::ap

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	ap
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> ap;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined ap (active pixel) 1 header field of an IEEE1722 stream with subtype RVF. The value range for ap shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the value of ap shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.3 colorspace

[SWS_RDS_90288] Definition of API variable ara::rds::IEEE1722DatagramRVF::colorspace

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	colorspace
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> colorspace;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined colorspace 8 bit header field of an IEEE1722 stream with subtype RVF. The value range for colorspace shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the value of colorspace shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.4 ef

[SWS_RDS_90281] Definition of API variable ara::rds::IEEE1722DatagramRVF::ef

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	ef
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> ef;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined ef (end frame) 1 bit header field of an IEEE1722 stream with subtype RVF. The value range for ef shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the value of ef shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.5 evt

[SWS_RDS_90282] Definition of API variable ara::rds::IEEE1722DatagramRVF::evt

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	evt
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> evt;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined evt 4 bit header field of an IEEE1722 stream with subtype RVF. The value range for evt shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the value of evt shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.6 f

[SWS_RDS_90280] Definition of API variable ara::rds::IEEE1722DatagramRVF::f

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	f
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> f;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined f (field) 1 bit header field of an IEEE1722 stream with subtype RVF. The value range for f shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the value of f shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.7 frame_rate

[SWS_RDS_90287] Definition of API variable ara::rds::IEEE1722DatagramRVF::frame_rate

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	frame_rate
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> frame_rate;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined frame_rate 8 bit header field of an IEEE1722 stream with subtype RVF. The value range for frame_rate shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, the value of frame_rate shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.8 i

[SWS_RDS_90284] Definition of API variable ara::rds::IEEE1722DatagramRVF::i

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	i
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> i;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined i (interlaced) 1 bit header field of an IEEE1722 stream with subtype RVF. The value range for i shall be: 0x00...0x01 : valid 0x02...0xFF : not used For transmission, the value of i shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.9 i_seq_num

[SWS_RDS_90290] Definition of API variable ara::rds::IEEE1722DatagramRVF::i_seq_num

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	i_seq_num
Type:	ara::core::Optional< std::uint16_t >
Syntax:	`ara::core::Optional<std::uint16_t> i_seq_num;`
Description:	std::uint16_t optional member variable which represents the IEEE1722 defined i_seq_num 8 bit header field of an IEEE1722 stream with subtype RVF. The value range for i_seq_num shall be: 0x00 00...0x00 FF : valid 0x01 00...0xFF FF : not used For transmission, the value of i_seq_num shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.10 line_number

[SWS_RDS_90291] Definition of API variable ara::rds::IEEE1722DatagramRVF::line_number

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	line_number
Type:	ara::core::Optional< std::uint32_t >
Syntax:	`ara::core::Optional<std::uint32_t> line_number;`
Description:	std::uint32_t optional member variable which represents the IEEE1722 defined line_number 16 bit header field of an IEEE1722 stream with subtype RVF. The value range for line_number shall be: 0x00 00 00 00 ... 0x00 00 FF FF : valid 0x00 01 00 00 ... 0xFF FF FF FF : not used For transmission, the value of line_number shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.11 num_lines

[SWS_RDS_90289] Definition of API variable ara::rds::IEEE1722DatagramRVF::num_lines

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	num_lines
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> num_lines;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined num_lines 4 bit header field of an IEEE1722 stream with subtype RVF. The value range for num_lines shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the value of num_lines shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.12 pd

[SWS_RDS_90283] Definition of API variable ara::rds::IEEE1722DatagramRVF::pd

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	pd
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> pd;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined pd (pull-down) 1 bit header field of an IEEE1722 stream with subtype RVF. The value range for pd shall be: 0x00...0x01 : valid 0x01...0xFF : not used For transmission, the value of pd shall be set to the configured value, if it is not provided by the application.
Visibility:	protected

⌋

8.3.8.1.13 pixel_depth

[SWS_RDS_90285] Definition of API variable ara::rds::IEEE1722DatagramRVF::pixel_depth

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	pixel_depth
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> pixel_depth;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined pixel_depth 4 bit header field of an IEEE1722 stream with subtype RVF. The value range for pixel_depth shall be: 0x00...0x0F : valid 0x01...0xFF : not used For transmission, the value of pixel_depth shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.14 pixel_format

[SWS_RDS_90286] Definition of API variable ara::rds::IEEE1722DatagramRVF::pixel_format

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	pixel_format
Type:	ara::core::Optional< std::uint8_t >
Syntax:	`ara::core::Optional<std::uint8_t> pixel_format;`
Description:	std::uint8_t optional member variable which represents the IEEE1722 defined pixel_format 4 bit header field of an IEEE1722 stream with subtype RVF. The value range for pixel_format shall be: 0x00...0x0F : valid 0x10...0xFF : not used For transmission, the value of pixel_format shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.1.15 total_lines

[SWS_RDS_90278] Definition of API variable ara::rds::IEEE1722DatagramRVF::total_lines

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	variable
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Symbol:	total_lines
Type:	ara::core::Optional< std::uint32_t >
Syntax:	`ara::core::Optional<std::uint32_t> total_lines;`
Description:	std::uint32_t optional member variable which represents the IEEE1722 defined total_lines 16 bit header field of an IEEE1722 stream with subtype RVF. The value range for total_lines shall be: 0x00 00 00 00 ... 0x00 00 FF FF : valid 0x00 01 00 00 ... 0xFF FF FF FF : not used For transmission, the value of total_lines shall be set to the configured value, if it is not provided the application.
Visibility:	protected

⌋

8.3.8.2 Public Member Functions

8.3.8.2.1 Special Member Functions

8.3.8.2.1.1 Destructor

[SWS_RDS_90292] Definition of API function ara::rds::IEEE1722DatagramRVF::~IEEE1722DatagramRVF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramRVF`
Syntax:	`~IEEE1722DatagramRVF () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramRVF that deletes the IEEE1722DatagramRVF instance.

⌋

8.3.9 Class: IEEE1722DatagramTSCF

[SWS_RDS_90274] Definition of API class ara::rds::IEEE1722DatagramTSCF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	IEEE1722DatagramTSCF
Base class:	IEEE1722DatagramAVTPDUCommonStreamHeaderFields
Syntax:	`class IEEE1722DatagramTSCF final : protected IEEE1722Datagram AVTPDUCommonStreamHeaderFields {...};`
Description:	This class defines an object which represents the IEEE1722 definded subtype TSCF header fields. Please note, the TSCF class do not define additional header fields, since all used header fields are covered by IEEE1722Datagram and IEEE1722DatagramAVTPDUCommonHeaderFields Data length (stream_data_length) and payload (data) is derived from class IEEE1722Datagram. Payload represents the IEEE1722 defined acf_payload_data.

⌋

8.3.9.1 Public Member Functions

8.3.9.1.1 Special Member Functions

8.3.9.1.1.1 Destructor

[SWS_RDS_90275] Definition of API function ara::rds::IEEE1722DatagramTSCF::~IEEE1722DatagramTSCF

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722DatagramTSCF`
Syntax:	`~IEEE1722DatagramTSCF () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722DatagramTSCF that deletes the IEEE1722DatagramTSCF instance.

⌋

8.3.10 Class: IEEE1722RawDataStreamConsumer

[SWS_RDS_90311] Definition of API class ara::rds::IEEE1722RawDataStreamConsumer

Status: DRAFT
Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00413

⌈

Kind:	class	
Port Interfaces:	`IEEE1722RawDataStreamConsumerInterface`	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Forwarding header file:	#include "ara/rds/rds_fwd.h"	
Scope:	namespace ara::rds	
Symbol:	IEEE1722RawDataStreamConsumer	
Syntax:	`template <class T>` `class IEEE1722RawDataStreamConsumer final {...};`	
Template param:	T	the type of the used IEEE1722 stream sub type (e.g. IEEE1722 AAF (audio) sub type)
Description:	This class defines a IEEE1722RawDataStreamConusmer object for consuming (reading) IEEE1722 streams from a network connection. .	

⌋

8.3.10.1 Public Member Functions

8.3.10.1.1 Special Member Functions

8.3.10.1.1.1 Copy Constructor

[SWS_RDS_90316] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::IEEE1722RawDataStreamConsumer

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00120, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`IEEE1722RawDataStreamConsumer (const IEEE1722RawDataStreamConsumer &)=delete;`
Description:	The copy constructor for IEEE1722RawDataStreamConsumer shall not be used.

⌋

8.3.10.1.1.2 Default Constructor

[SWS_RDS_90313] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::IEEE1722RawDataStreamConsumer

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00120, RS_AP_00129, RS_AP_00146

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`IEEE1722RawDataStreamConsumer ()=delete;`
Description:	The default constructor for IEEE1722RawDataStreamConsumer shall not be used.

⌋

8.3.10.1.1.3 Move Constructor

[SWS_RDS_90314] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::IEEE1722RawDataStreamConsumer

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`IEEE1722RawDataStreamConsumer (IEEE1722RawDataStreamConsumer &&rdsc)=delete;`
Description:	The move constructor for IEEE1722RawDataStreamConsumer shall not be used.

⌋

8.3.10.1.1.4 Copy Assignment Operator

[SWS_RDS_90317] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::operator=

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00119, RS_AP_00120, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`IEEE1722RawDataStreamConsumer & operator= (const IEEE1722RawDataStream Consumer &)=delete;`
Description:	The copy assignment operator for IEEE1722RawDataStreamConsumer shall not be used.

⌋

8.3.10.1.1.5 Move Assignment Operator

[SWS_RDS_90315] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::operator=

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`IEEE1722RawDataStreamConsumer & operator= (IEEE1722RawDataStream Consumer &&rdsc)=delete;`
Description:	The move assignment operator for IEEE1722RawDataStreamConsumer shall not be used.

⌋

8.3.10.1.1.6 Destructor

[SWS_RDS_90318] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::~IEEE1722RawDataStreamConsumer

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`~IEEE1722RawDataStreamConsumer () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722RawDataStreamConsumer that deletes the IEEE1722RawData StreamConsumer instance. If the intance is still connected to corresponding data link layer socket, the connection is closed and shut down (calling Shutdown()) before destroying the IEEE1722RawDataStreamConsumer object.

⌋

8.3.10.1.2 Member Functions

8.3.10.1.2.1 Connect

[SWS_RDS_90319] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::Connect

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`	
Syntax:	`ara::core::Result< void > Connect () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kConnection Refused	rollback_semantics
		The target address was not listening for connections or refused the connection request.
	RdsErrc::kStream AlreadyConnected	rollback_semantics
		The specified connection is already connected.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Sets up a data link socket connection for the IEEE1722RawDataStreamConsumer defined by the instance, and establishes internal connection. If MACSec is configured for the data link socket connection, the MACSec connection shall be initialized here.	

⌋

8.3.10.1.2.2 ReadData

[SWS_RDS_90321] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::ReadData

Status: DRAFT
Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`
Syntax:	`ara::core::Result< ara::core::Vector< T > > ReadData (std::size_t maxNumberOfDatagrams) noexcept;`

▽

△

Parameters (in):	maxNumberOf Datagrams	The requested maximum number of datagrams to read from the stream.
Return value:	ara::core::Result< ara::core::Vector< T > >	A Result with an Vector of IEEE1722 datagrams if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
	RdsErrc::kStreamHeader FieldValueInvalid	rollback_semantics
		Detected an invalid stream header field value(e.g. unkown IEEE1722 stream id).
Description:	Requests to read a number of IEEE1722 datagrams from the data link socket connection for the IEEE1722 stream defined by the instance.	

⌋

8.3.10.1.2.3 Shutdown

[SWS_RDS_90320] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::Shutdown

Status: DRAFT

Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`	
Syntax:	`ara::core::Result< void > Shutdown () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Closes the data link socket connection for this IEEE1722 stream consumer instance. Receiving from the data link socket connection shall shut down.	

⌋

8.3.10.1.3 Named Constructors

8.3.10.1.3.1 Create

[SWS_RDS_90312] Definition of API function ara::rds::IEEE1722RawDataStream Consumer::Create

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamConsumer`	
Syntax:	`static ara::core::Result< IEEE1722RawDataStreamConsumer > Create (const ara::core::InstanceSpecifier &instance) noexcept;`	
Parameters (in):	instance	The instance specifier for the instance.
Return value:	ara::core::Result< IEEE1722RawData StreamConsumer >	ara::core::Result<IEEE1722RawDataStreamConsumer>, the IEEE1722RawDataStreamConsumer object if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	RdsErrc::kConnection CreationFailed	rollback_semantics
		Permission to create a connection is denied. (POSIX EACCES)
	RdsErrc::kStream AlreadyConnected	rollback_semantics
		The specified connection is already connected.
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of the executable, or it refers to a model element other than a PortPrototype.
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `RawDataStreamMapping` needs to be typed by a `IEEE1722RawDataStreamConsumerInterface`
	`ProcessMappingViolation`	The type of mapping does not match the expected type of Port Interface
	`InstanceSpecifierAlreadyInUseViolation`	The constructor was called with an Instance Specifier already in use in this process
Description:	Named exception-less constructor that takes an instance Specifier qualifying the use IEEE1722 specified stream id for the instance. A data link layer socket shall be created, which used to receive IEEE1722 streams via the network.	

⌋

8.3.11 Class: IEEE1722RawDataStreamProducer

[SWS_RDS_90322] Definition of API class ara::rds::IEEE1722RawDataStream Producer

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	class	
Port Interfaces:	`IEEE1722RawDataStreamProducerInterface`	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Forwarding header file:	#include "ara/rds/rds_fwd.h"	
Scope:	namespace ara::rds	
Symbol:	IEEE1722RawDataStreamProducer	
Syntax:	`template <class T>` `class IEEE1722RawDataStreamProducer final {...};`	
Template param:	T	the type of the used IEEE1722 stream sub type (e.g. IEEE1722 AAF (audio) sub type)
Description:	This class defines a IEEE1722RawDataStreamProducer object for producing (writing) data via an IEEE1722 streams to a network connection. .	

⌋

8.3.11.1 Public Member Functions

8.3.11.1.1 Special Member Functions

8.3.11.1.1.1 Copy Constructor

[SWS_RDS_90327] Definition of API function ara::rds::IEEE1722RawDataStream Producer::IEEE1722RawDataStreamProducer

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00120, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`
Syntax:	`IEEE1722RawDataStreamProducer (const IEEE1722RawDataStreamProducer &)=delete;`
Description:	The copy constructor for IEEE1722RawDataStreamProducer shall not be used.

⌋

8.3.11.1.1.2 Default Constructor

[SWS_RDS_90324] Definition of API function ara::rds::IEEE1722RawDataStream Producer::IEEE1722RawDataStreamProducer

Status: DRAFT

Upstream requirements: RS_CM_00413, RS_AP_00120, RS_AP_00129, RS_AP_00146

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`
Syntax:	`IEEE1722RawDataStreamProducer ()=delete;`
Description:	The default constructor for IEEE1722RawDataStreamProducer shall not be used.

⌋

8.3.11.1.1.3 Move Constructor

[SWS_RDS_90325] Definition of API function ara::rds::IEEE1722RawDataStream Producer::IEEE1722RawDataStreamProducer

Status: DRAFT

Upstream requirements: RS_CM_00413, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`
Syntax:	`IEEE1722RawDataStreamProducer (IEEE1722RawDataStreamProducer &&rdsp)=delete;`
Description:	The move constructor for IEEE1722RawDataStreamProducer shall not be used.

⌋

8.3.11.1.1.4 Copy Assignment Operator

[SWS_RDS_90328] Definition of API function ara::rds::IEEE1722RawDataStream Producer::operator=

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00119, RS_AP_00120, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`
Syntax:	`IEEE1722RawDataStreamProducer & operator= (const IEEE1722RawDataStream Producer &)=delete;`
Description:	The copy assignment operator for IEEE1722RawDataStreamProducer shall not be used.

⌋

8.3.11.1.1.5 Move Assignment Operator

[SWS_RDS_90326] Definition of API function ara::rds::IEEE1722RawDataStream Producer::operator=

Status: DRAFT
Upstream requirements: RS_CM_00413, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`
Syntax:	`IEEE1722RawDataStreamProducer & operator= (IEEE1722RawDataStream Producer &&rdsp)=delete;`
Description:	The move assignment operator for IEEE1722RawDataStreamProducer shall not be used.

⌋

8.3.11.1.1.6 Destructor

[SWS_RDS_90329] Definition of API function ara::rds::IEEE1722RawDataStream Producer::~IEEE1722RawDataStreamProducer

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`
Syntax:	`~IEEE1722RawDataStreamProducer () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor of the IEEE1722RawDataStreamProducer that deletes the IEEE1722RawData StreamProducer instance. If the connection is still open, the connection is closed and shut down (calling Shutdown()) before destroying the IEEE1722RawDataStreamProducer object.

⌋

8.3.11.1.2 Member Functions

8.3.11.1.2.1 Connect

[SWS_RDS_90330] Definition of API function ara::rds::IEEE1722RawDataStream Producer::Connect

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`	
Syntax:	`ara::core::Result< void > Connect () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kConnection Refused	rollback_semantics
		The target address was not listening for connections or refused the connection request.
	RdsErrc::kStream AlreadyConnected	rollback_semantics
		The specified connection is already connected.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics

▽

△

		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Sets up a data link socket connection for the IEEE1722RawDataStreamProducer defined by the instance and establishes an internal connection. If MACSec is configured for the data link socket connection, the MACSec connection shall be initialized here.	

⌋

8.3.11.1.2.2 Shutdown

[SWS_RDS_90331] Definition of API function ara::rds::IEEE1722RawDataStream Producer::Shutdown

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`	
Syntax:	`ara::core::Result< void > Shutdown () noexcept;`	
Return value:	ara::core::Result< void >	void if successful, otherwise an error code indicating the error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
Description:	Closes the data link socket connection for the IEEE1722 stream defined by the instance. Transmission via the data link socket connection shall shut down.	

⌋

8.3.11.1.2.3 WriteData

[SWS_RDS_90332] Definition of API function ara::rds::IEEE1722RawDataStream Producer::WriteData

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`	
Syntax:	`ara::core::Result< std::size_t > WriteData (ara::core::Vector< T > data) noexcept;`	
Parameters (in):	data	Vector of IEEE1722 datagrams.
Return value:	ara::core::Result< std::size_t >	A Result of size_t to indicate the number of valid datagrams which were writen successfully, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	RdsErrc::kStreamNot Connected	rollback_semantics
		Trying to use a raw data stream without an established connection.
	RdsErrc::kInterruptedBy Signal	no_rollback_semantics
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
	RdsErrc::kStreamHeader FieldValueInvalid	rollback_semantics
		Detected an invalid stream header field value(e.g. unkown IEEE1722 stream id).
	RdsErrc::kStreamHeader FieldValueMissing	rollback_semantics
		Decected an missing stream header field (e.g. missing IEEE1722 mac address part of stream id).
Description:	Requests to write a number of datagrams to the data link socket connection for the IEEE1722 stream defined by the instance.	

⌋

8.3.11.1.3 Named Constructors

8.3.11.1.3.1 Create

[SWS_RDS_90323] Definition of API function ara::rds::IEEE1722RawDataStream Producer::Create

Status: DRAFT
Upstream requirements: RS_CM_00413

⌈

Kind:	function	
Header file:	#include "ara/rds/raw_data_stream_IEEE1722.h"	
Scope:	`class ara::rds::IEEE1722RawDataStreamProducer`	
Syntax:	`static ara::core::Result< IEEE1722RawDataStreamProducer > Create (const ara::core::InstanceSpecifier &instance) noexcept;`	
Parameters (in):	instance	The instance specifier for the instance.
Return value:	ara::core::Result< IEEE1722RawData StreamProducer >	ara::core::Result<IEEE1722RawDataStreamProducer>, the IEEE1722RawDataStreamProducer object if succesful, otherwise an error code indicating the error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	RdsErrc::kConnection CreationFailed	rollback_semantics
		Permission to create a connection is denied. (POSIX EACCES)
	RdsErrc::kStream AlreadyConnected	rollback_semantics
		The specified connection is already connected.
Violations:	`InstanceSpecifierMappingIntegrityViolation`	InstanceSpecifier either cannot be resolved in the model in the context of the executable, or it refers to a model element other than a PortPrototype.
	`PortInterfaceMappingViolation`	A `PortPrototype` that is referenced by a `RawDataStreamMapping` needs to be typed by a `IEEE1722RawDataStreamProducerInterface`
	`ProcessMappingViolation`	The type of mapping does not match the expected type of Port Interface
	`InstanceSpecifierAlreadyInUseViolation`	The constructor was called with an Instance Specifier already in use in this process
Description:	Named exception-less constructor that takes an instance specifier qualifying the used IEEE1722 specified stream id for the instance. A data link layer socket shall be created, which used to transmit IEEE1722 streams via the network.	

⌋

8.4 Header: ara/rds/rds_error_domain.h

8.4.1 Non-Member Types

8.4.1.1 Enumeration: RdsErrc

[SWS_RDS_12367] Definition of API enum ara::rds::RdsErrc

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	enumeration	
Header file:	#include "ara/rds/rds_error_domain.h"	
Forwarding header file:	#include "ara/rds/rds_fwd.h"	
Scope:	namespace ara::rds	
Symbol:	RdsErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class RdsErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kStreamNotConnected	= 1
		Trying to use a raw data stream without an established connection.
	kCommunicationTimeout	= 2
		The operation was not successful and timed out.
	kConnectionRefused	= 3
		The target address was not listening for connections or refused the connection request.
	kAddressNotAvailable	= 4
		The specified address is not available from the local machine.
	kStreamAlready Connected	= 5
		The specified connection is already connected.
	kConnectionClosedBy Peer	= 6
		Network error. The established connection has been shut down during writing (POSIX EPIPE).
	kPeerUnreachable	= 7
		Network error. The peer is unreachable (POSIX ENETUNREACH).
	kConnectionAborted	= 8
		Network error. The incoming connection was aborted (POSIX ECONNABORTED).
	kInterruptedBySignal	= 9
		The operation was interrupted by a system signal (POSIX_EINTR). Usually a retry of the operation works, but resources may have been put into an unknown state, which means that retrying might lead to unexpected behavior.
	kConnectionCreation Failed	= 10
		Permission to create a connection is denied. (POSIX EACCES)
	kStreamHeaderField ValueInvalid	= 13
		Detected an invalid stream header field value(e.g. unkown IEEE1722 stream id).

▽

△

	kStreamHeaderField ValueMissing	= 14
		Decected an missing stream header field (e.g. missing IEEE1722 mac address part of stream id).
Description:	The RdsErrc enumeration defines the error codes for the RdsErrorDomain.	

⌋

8.4.2 Non-Member Functions

8.4.2.1 Other

8.4.2.1.1 GetRdsErrorDomain

[SWS_RDS_11298] Definition of API function ara::rds::GetRdsErrorDomain

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/rds/rds_error_domain.h"	
Scope:	namespace ara::rds	
Syntax:	`constexpr ara::core::ErrorDomain & GetRdsErrorDomain () noexcept;`	
Return value:	ara::core::ErrorDomain &	A reference to the global `ara::rds::RdsErrorDomain` object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns a reference to the global `ara::rds::RdsErrorDomain` object.	

⌋

8.4.2.1.2 MakeErrorCode

[SWS_RDS_11299] Definition of API function ara::rds::MakeErrorCode

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/rds/rds_error_domain.h"	
Scope:	namespace ara::rds	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (ara::rds::RdsErrc 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 `ara::core::ErrorCode` object.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe
Description:	Creates an instance of `ara::core::ErrorCode`.

⌋

8.4.3 Class: RdsErrorDomain

[SWS_RDS_11293] Definition of API class ara::rds::RdsErrorDomain

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	class
Header file:	#include "ara/rds/rds_error_domain.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	RdsErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class RdsErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::rds::RdsErrorDomain` in [SWS_CORE_90023]
Description:	Defines a class representing the Raw Data Streams error domain.

⌋

8.4.3.1 Public Member Types

8.4.3.1.1 Type Alias: Errc

[SWS_RDS_11326] Definition of API type ara::rds::RdsErrorDomain::Errc

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	type alias
Header file:	#include "ara/rds/rds_error_domain.h"
Scope:	`class ara::rds::RdsErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = RdsErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.4.3.1.2 Type Alias: Exception

[SWS_RDS_11327] Definition of API type ara::rds::RdsErrorDomain::Exception

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	type alias
Header file:	#include "ara/rds/rds_error_domain.h"
Scope:	`class ara::rds::RdsErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = RdsException;`
Description:	Alias for the exception base class.

⌋

8.4.3.2 Public Member Functions

8.4.3.2.1 Special Member Functions

8.4.3.2.1.1 Default Constructor

[SWS_RDS_11294] Definition of API function ara::rds::RdsErrorDomain::RdsErrorDomain

Upstream requirements: RS_AP_00120, RS_AP_00130, RS_AP_00146

⌈

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

⌋

8.4.3.2.2 Member Functions

8.4.3.2.2.1 Message

[SWS_RDS_11296] Definition of API function ara::rds::RdsErrorDomain::Message

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/rds/rds_error_domain.h"	
Scope:	`class ara::rds::RdsErrorDomain`	
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.	

⌋

8.4.3.2.2.2 Name

[SWS_RDS_11295] Definition of API function ara::rds::RdsErrorDomain::Name

Upstream requirements: RS_AP_00120, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/rds/rds_error_domain.h"	
Scope:	`class ara::rds::RdsErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	As per `ara::rds::RdsErrorDomain` in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns a string constant associated with RdsErrorDomain.	

⌋

8.4.3.2.2.3 ThrowAsException

[SWS_RDS_11297] Definition of API function ara::rds::RdsErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/rds/rds_error_domain.h"	
Scope:	`class ara::rds::RdsErrorDomain`	
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 RdsException 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.	

⌋

8.4.4 Class: RdsException

[SWS_RDS_11291] Definition of API class ara::rds::RdsException

Upstream requirements: RS_AP_00130, RS_AP_00122, RS_AP_00127

⌈

Kind:	class
Header file:	#include "ara/rds/rds_error_domain.h"
Forwarding header file:	#include "ara/rds/rds_fwd.h"
Scope:	namespace ara::rds
Symbol:	RdsException
Base class:	ara::core::Exception
Syntax:	`class RdsException : public ara::core::Exception {...};`
Description:	Defines a class for exceptions to be thrown by the Raw Data Streams.

⌋

8.4.4.1 Public Member Functions

8.4.4.1.1 Constructors

8.4.4.1.1.1 RdsException

[SWS_RDS_11292] Definition of API function ara::rds::RdsException::RdsException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00130

⌈

Kind:	function	
Header file:	#include "ara/rds/rds_error_domain.h"	
Scope:	`class ara::rds::RdsException`	
Syntax:	`explicit RdsException (ara::core::ErrorCode errorCode) noexcept;`	
Parameters (in):	errorCode	The error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constructs a new RdsException 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 [8]. This chapter defines the default values for attributes and semantic constraints for elements specified in [8] that are part of the configuration model of this functional cluster.

The configuration structure of the Ethernet socket connections in the Raw Data Stream functional cluster is described in [8] as part of the Service Instance Manifest.

This chapter describes how to apply the configuration parameters and credentials for RawDataStreamClients and RawDataStreamServers.

[SWS_RDS_99004] Ethernet endpoint configuration

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈

Ethernet socket connections are statically configured in the Deployment model as part of the Service Instance Manifest, and used throughout the connected session for the RawDataStreams communication. The following configuration elements can be specified on the Deployment model of each RawDataStreamClient or RawDataStream-Server instance, identified through the InstanceSpecifier provided to the constructor.

RawDataStreamClient endpoint and credentials configuration elements:

- Local Network Endpoint: `EthernetRawDataStreamClientMapping.localCommConnector`
- Local UdpPort: `EthernetRawDataStreamClientMapping.localUdpPort`
- Local TcpPort: `EthernetRawDataStreamClientMapping.localTcpPort`
- Socket Options: `EthernetRawDataStreamClientMapping.socketOption`
- (D)TLS properties: `EthernetRawDataStreamClientMapping.tlsSecureComProps`
- Remote Unicast Credentials: `EthernetRawDataStreamClientMapping.unicastCredentials` (UDP/TCP)
- Multicast Credentials: `EthernetRawDataStreamClientMapping.multicastCredentials` (UDP only)

RawDataStreamServer endpoint and credentials configuration elements:

- Local Network Endpoint: `EthernetRawDataStreamServerMapping.localCommConnector`
- Local UdpPort: `EthernetRawDataStreamServerMapping.localUdpPort`
- Local TcpPort: `EthernetRawDataStreamServerMapping.localTcpPort`
- Socket Options: `EthernetRawDataStreamServerMapping.socketOption`

- (D)TLS properties: EthernetRawDataStreamServerMapping.tlsSecureComProps
- Remote Unicast Credentials: EthernetRawDataStreamServerMapping.unicastUdpCredentials (UDP only)
- Multicast Credentials: EthernetRawDataStreamServerMapping.multicastCredentials (UDP only)

For the RawDataStreamClients the following shall apply:

- Remote server credentials for unicast communication must always be defined for the client. The Unicast remote server credentials are configured in RawDataStreamEthernetTcpUdpCredentials aggregated by the EthernetRawDataStreamClientMapping in the role unicastCredentials.
- A tcpPort and udpPort shall not be defined in the same RawDataStreamEthernetTcpUdpCredentials element.
- If a TcpPort is defined in the EthernetRawDataStreamClientMapping.unicastCredentials, these credentials are used for Connect() calls to establish the connection to the server.
- This unicast connection shall always be used for WriteData() calls to send data to the server (for both UDP and TCP).
- If Multicast Credentials are defined for the client, the RawDataStream shall bind and join the multicast address and udpPort given in the MulticastCredentials. The MulticastCredentials is configured in RawDataStreamEthernetUdpCredentials aggregated by the EthernetRawDataStreamClientMapping. This multicast socket connection shall be read from when ReadData() is called.
- If no MulticastCredentials are defined for the client, the Unicast Remote Credentials shall also be used for ReadData() calls.

For the RawDataStreamServers the following shall apply:

- If Multicast Credentials is defined for the server, a multicast connection shall be created using the Multicast Credentials which are configured in RawDataStreamEthernetUdpCredentials aggregated by the EthernetRawDataStreamServerMapping in the role multicastCredentials. Then the data is sent on this multicast socket when WriteData() is called.
- If Remote Unicast Credentials are defined for the server, a unicast socket shall be created using the Unicast Credentials which are configured in RawDataStreamEthernetUdpCredentials aggregated by the EthernetRawDataStreamServerMapping in the role unicastUdpCredentials. Then the data is sent on this unicast socket when WriteData() is called.
- The local credentials defined in EthernetCommunicationConnector shall always be used to create a unicast socket and read data from a client when ReadData() is called on the server side. If no local credentials are defined, reading of

data from the server cannot be performed, and an error kStreamNotConnected will be returned.

- If a `localTcpPort` is defined in `EthernetRawDataStreamServerMapping`, the credentials defined in `EthernetCommunicationConnector` are used to create, bind, and listen to the socket used for TCP communication when the constructor of RawDataStream is called. Then the server accepts incoming connection requests when WaitForConnection() is called.

⌋

[SWS_RDS_90216] Socket Options configuration

Upstream requirements: RS_CM_00410, RS_CM_00411, RS_CM_00412

⌈For both RawDataStreamClients and RawDataStreamServers a list of socket options can be defined in the attribute `socketOption` to be applied to the sockets created for unicast or multicast communication. The options shall be specified as a list of strings. The accepted values are platform specific and shall be documented by the vendor.⌋

An example of `socketOption` definition is to provide a series of "option", "value" pairs for POSIX socket level options, e.g.: ["SO_KEEPALIVE", "1", SO_RCVBUF","1024"]

[SWS_RDS_90217] TLS properties configuration

Upstream requirements: RS_CM_00410, RS_CM_00411

⌈For both RawDataStreamClients and RawDataStreamServers (D)TLS properties can be defined in the attributes `tlsSecureComProps` to configure usage of TLS to create secure UDP and TCP channels for the RawDataStreams according to the Transport Layer Security protocol. See [SWS_RDS_90211]⌋

Note: Usage of (D)TLS is restricted to 1:1 socket connections (use case 1 and 2 of figure Figure 7.2).

10.1 Default Values

This functional cluster does not define any default values for attributes specified in [8].

10.2 Semantic Constraints

This section defines semantic constraints for elements specified in [8] that are part of the configuration model of this functional cluster.

[SWS_RDS_CONSTR_00001] Configurable Namespace for RawDataStream ⌈`AbstractRawDataStreamInterface`.namespace shall never exist.⌋

A Mentioned Manifest Elements

This chapter contains the remaining set of meta-class tables which are not shown directly in the main body of this document.

This chapter is generated.

Class	***AbstractRawDataStreamEthernetCredentials*** (abstract)			
Note	This meta-class serves as an abstract base class for the configuration of network credentials. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *Describable*			
Subclasses	RawDataStreamEthernetTcpUdpCredentials, RawDataStreamEthernetUdpCredentials			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipV4Address	Ip4AddressString	0..1	attr	This attribute describes the IP V4 address of the remote server.
ipV6Address	Ip6AddressString	0..1	attr	This attribute describes the IP V6 address of the remote server.
udpPort	PositiveInteger	0..1	attr	This attribute represents the configuration of a UDP port number.

Table A.1: AbstractRawDataStreamEthernetCredentials

Class	***AbstractRawDataStreamInterface*** (abstract)			
Note	This meta-class serves as an abstract base class for PortInterfaces related to raw data streams. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*MacLayerRawDataStreamInterface*, RawDataStreamClientInterface, RawDataStreamServerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.2: AbstractRawDataStreamInterface

Class	***CommunicationConnector*** (abstract)			
Note	The connection between the referencing ECU and the referenced channel via the referenced controller. Connectors are used to describe the bus interfaces of the ECUs and to specify the sending/receiving behavior. Each CommunicationConnector has a reference to exactly one communicationController. Note: Several CommunicationConnectors can be assigned to one PhysicalChannel in the scope of one ECU Instance.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationConnector*, EthernetCommunicationConnector, FlexrayCommunication Connector, UserDefinedCommunicationConnector			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***CommunicationConnector*** (abstract)			
commController	Communication Controller	0..1	ref	Reference to the communication controller. The CommunicationConnector and referenced CommunicationController shall be aggregated by the same ECUInstance. The communicationController can be referenced by several CommunicationConnector elements. This is important for the FlexRay Bus. FlexRay communicates via two physical channels. But only one controller in an ECU is responsible for both channels. Thus, two connectors (for channel A and for channel B) shall reference to the same controller.
createEcu WakeupSource	Boolean	0..1	attr	If this parameter is available and set to true then a channel wakeup source shall be created for the Physical Channel referencing this CommunicationConnector.
explicitWakeup Channel	PhysicalChannel	*	ref	Defines a restriction which PhysicalChannels shall be woken up if this CommunicationConnector is the wakeup source. If not defined, then no restriction applies. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=explicitWakeupChannel.physicalChannel, explicitWakeupChannel.variationPoint.shortLabel vh.latestBindingTime=postBuild
explicitWakeup Pnc	PncMappingIdent	*	ref	Defines a restriction which PNCs shall be woken up if this CommunicationConnector is the wakeup source. If not defined, then no restriction applies. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=explicitWakeupPnc.pncMappingIdent, explicitWakeupPnc.variationPoint.shortLabel vh.latestBindingTime=postBuild
pncFilterArray Mask (ordered)	PositiveInteger	*	attr	Bit mask for NM-Pdu Payload used to configure the NM filter mask for the Network Management.

Table A.3: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Note	Ethernet specific attributes to the CommunicationConnector.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apApplication Endpoint	ApApplicationEndpoint	*	aggr	Collection of Application Addresses that are used on the CommunicationConnector. This Attribute is only used by the AUTOSAR Adaptive Platform.
canXlProps	CanXlProps	*	ref	If the Ethernet frames handled by this Ethernet CommunicationConnector are tunneled through CAN XL, then this reference shall refer the CanXlProps which contains the specific configuration parameters of the CAN XL controller of the physical CAN XL connection to be used for tunneling. This Attribute is only used by the AUTOSAR Adaptive Platform.
maximum Transmission Unit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
neighborCache Size	PositiveInteger	0..1	attr	This attribute specifies the size of neighbor cache or ARP table in units of entries.

▽

△

Class	**EthernetCommunicationConnector**			
pathMtu Enabled	Boolean	0..1	attr	If enabled the IPv4/IPv6 processes incoming ICMP "Packet Too Big" messages and stores a MTU value for each destination address. **Tags:** atp.Status=obsolete
pathMtuTimeout	TimeValue	0..1	attr	If this value is >0 the IPv4/IPv6 will reset the MTU value stored for each destination after n seconds. **Tags:** atp.Status=obsolete
unicastNetwork Endpoint	NetworkEndpoint	*	ref	Network Endpoint that defines the IPAddress of the machine. This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.4: EthernetCommunicationConnector

Class	«atpVariation» **EthernetCommunicationController**			
Note	Ethernet specific communication port attributes.			
Base	*ARObject*, *CommunicationController*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canXlConfig	AbstractCan Communication Controller	0..1	ref	If the Ethernet frames handled by this Ethernet CommunicationController are to be tunneled through CAN XL, then this reference shall refer to the Abstract CanCommunicationController that aggregates the Can ControllerXlConfiguration of the physical CAN XL channel to be used for tunneling.
couplingPort	CouplingPort	*	aggr	Optional CouplingPort that can be used to connect the ECU to a CouplingElement (e.g. a switch).
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the ethernet controller.
macUnicast Address	MacAddressString	0..1	attr	Media Access Control address (MAC address) that uniquely identifies each EthernetCommunication Controller in the network.
maximum ReceiveBuffer Length	Integer	0..1	attr	Determines the maximum receive buffer length (frame length) in bytes.
maximum TransmitBuffer Length	Integer	0..1	attr	Determines the maximum transmit buffer length (frame length) in bytes.
slaveActAs Passive Communication Slave	Boolean	0..1	attr	This attribute specifies if the EcuInstance is acting as a passive communication slave on the connected Physical Channel. This is used for EthernetCommunication Controllers that use Ethernet hardware which supports wake-up and sleep on the network (e.g. Open Alliance TC10 compliant Ethernet hardware).
slaveQualified UnexpectedLink DownTime	TimeValue	0..1	attr	This attribute specifies time when an unexpected link down is evaluated as link down and indicated to the AUTOSAR communication stack.

Table A.5: EthernetCommunicationController

Class	***EthernetMacRawDataStreamMapping*** (abstract)
Note	This meta-class serves as the abstract bases class for the ability to map a PortPrototype to an Ethernet-Mac-layer based communication. **Tags:** atp.Status=candidate This Class is only used by the AUTOSAR Adaptive Platform.

▽

△

Class	***EthernetMacRawDataStreamMapping*** (abstract)			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*IEEE1722RawDataStreamMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localComm Connector	EthernetCommunication Connector	0..1	ref	This reference represents the CommunicationConnector taken for Mac-based data communication. **Tags:** atp.Status=candidate
socketOption	String	*	attr	This attribute represents the ability to specify non-formal socket options that might only be valid for specific platforms. AUTOSAR does not define a standardized meaning for the possible values of this attribute. **Tags:** atp.Status=candidate

Table A.6: EthernetMacRawDataStreamMapping

Class	**EthernetRawDataStreamClientMapping**			
Note	This meta-class represents the ability to map a client PortPrototype to a Ethernet-based communication channel. **Tags:** atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteServer Config	EthernetRawData StreamRemoteServer Config	0..1	aggr	This aggregation is used to configure the credentials of the remote server.

Table A.7: EthernetRawDataStreamClientMapping

Class	**EthernetRawDataStreamLocalEndpointConfig**			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote endpoint in the context of a raw data stream mapping. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	*EthernetRawDataStreamMapping*.localEndpointConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localComm Connector	EthernetCommunication Connector	0..1	ref	This attribute represents the CommunicationConnector taken for socket-based data communication.
localTcpPort	ApApplicationEndpoint	0..1	ref	This aggregation represents the configuration of a local TCP port number.
localUdpPort	ApApplicationEndpoint	0..1	ref	This aggregation represents the configuration of a local unicast UDP port number.

Table A.8: EthernetRawDataStreamLocalEndpointConfig

Class	***EthernetRawDataStreamMapping*** (abstract)			
Note	This meta-class serves as the abstract bases class for the ability to map a PortPrototype to a Ethernet-based communication channel. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	EthernetRawDataStreamClientMapping, EthernetRawDataStreamServerMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localEndpoint Config	EthernetRawData StreamLocalEndpoint Config	0..1	aggr	This aggregation is used to configure the credentials of the endpoint.
socketOption	String	*	attr	This attribute represents the ability to specify non-formal socket options that might only be valid for specific platforms. AUTOSAR does not define a standardized meaning for the possible values of this attribute.
tlsSecureCom Props	TlsSecureComProps	0..1	ref	This reference provides the ability to define TLS-related properties for the enclosing SocketRawDataStream Mapping.

Table A.9: EthernetRawDataStreamMapping

Class	**EthernetRawDataStreamRemoteClientConfig**			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote server in the context of a raw data stream client mapping. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	EthernetRawDataStreamServerMapping.remoteClientConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicast Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of multicast credentials for communication with a remote raw data stream client.
unicastUdp Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of a remote raw data stream client that communicates via unicast over UDP.

Table A.10: EthernetRawDataStreamRemoteClientConfig

Class	**EthernetRawDataStreamRemoteServerConfig**			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote server in the context of a raw data stream client mapping. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*			
Aggregated by	EthernetRawDataStreamClientMapping.remoteServerConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicast Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of multicast credentials for communication with a remote raw data stream server.
unicast Credentials	RawDataStream EthernetTcpUdp Credentials	0..1	aggr	This meta-class represents the ability to map a server PortPrototype to a Ethernet-based communication channel.

Table A.11: EthernetRawDataStreamRemoteServerConfig

Class	**EthernetRawDataStreamServerMapping**			
Note	This meta-class represents the ability to map a server PortPrototype to a Ethernet-based communication channel. **Tags:** atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteClient Config	EthernetRawData StreamRemoteClient Config	0..1	aggr	This aggregation is used to configure the credentials of the remote client.

Table A.12: EthernetRawDataStreamServerMapping

Class	**GlobalTimeDomain**			
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. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild
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
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

△

Class	**GlobalTimeDomain**			
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.13: GlobalTimeDomain

Class	**IEEE1722RawDataStreamConsumerInterface**			
Note	This meta-class represents the necessary capabilities for IEEE1722 raw data streaming on the consumer side, i.e. the streaming of data that do not undergo any serialization. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MacLayerRawDataStreamInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.14: IEEE1722RawDataStreamConsumerInterface

Class	**IEEE1722RawDataStreamConsumerMapping**			
Note	This meta-class represents the ability to map a consumer PortPrototype to an Ethernet-Mac-layer IEEE1722 based communication. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *IEEE1722RawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ieee1722Stream	IEEE1722TpConnection	0..1	ref	Reference to the IEEE1722TpConnection. **Tags:** atp.Status=candidate

Table A.15: IEEE1722RawDataStreamConsumerMapping

Class	**IEEE1722RawDataStreamProducerInterface**			
Note	This meta-class represents the necessary capabilities for IEEE1722 raw data streaming on the producer side, i.e. the streaming of data that do not undergo any serialization. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MacLayerRawDataStreamInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.16: IEEE1722RawDataStreamProducerInterface

Class	**IEEE1722RawDataStreamProducerMapping**			
Note	This meta-class represents the ability to map a producer PortPrototype to an Ethernet-Mac-layer IEEE1722 based communication. **Tags:** atp.Status=candidate atp.recommendedPackage=RawDataStreamingMappings This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetMacRawDataStreamMapping*, *IEEE1722RawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ieee1722Stream	IEEE1722TpConnection	0..1	ref	Reference to the IEEE1722TpConnection. **Tags:** atp.Status=candidate

Table A.17: IEEE1722RawDataStreamProducerMapping

Class	**IEEE1722TpAcfConnection**
Note	ACF IEEE1722Tp connection. **Tags:** atp.Status=candidate atp.recommendedPackage=IEEE1722TpConnections

▽

△

Class	**IEEE1722TpAcfConnection**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *IEEE1722TpConnection*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
acfMaxTransitTime	TimeValue	0..1	attr	Defines the time offset that is added to the current time at the producer in order to get the "presentation time" (in seconds) when content shall be presented at the consumers.
acfTransportedBus	IEEE1722TpAcfBus	*	aggr	Definition of the transported busses over this ACF connection. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=acfTransportedBus.shortName, acfTransportedBus.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
collectionThreshold	PositiveInteger	0..1	attr	Defines the size threshold in bytes which, when exceeded, triggers the sending of the IEEE1722Tp ACF message, even when the maxium IEEE1722Tp ACF message size has not been reached yet.
collectionTimeout	TimeValue	0..1	attr	When this timeout expires the IEEE1722Tp ACF message is triggered for sending. The respective timer is started when the first Pdu is put into the IEEE1722Tp ACF message. Defined in seconds.
mixedBusTypeCollection	Boolean	0..1	attr	Defines if this ACF-stream is allowed to collect ACF-messages of different bus kinds (i.e. whether it is allowed to collect CAN and LIN ACF-messages in one ACF-stream message).

Table A.18: IEEE1722TpAcfConnection

Class	***IEEE1722TpAvConnection*** (abstract)			
Note	AV IEEE1722Tp connection. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *IEEE1722TpConnection*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	IEEE1722TpAafConnection, IEEE1722TpCrfConnection, IEEE1722TplidcConnection, IEEE1722TpRvfConnection			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maxTransitTime	TimeValue	0..1	attr	Defines the time offset that is added to the current time at the producer in order to get the "presentation time" (in seconds) when content shall be presented at the consumers.

Table A.19: IEEE1722TpAvConnection

Class	***IEEE1722TpConnection*** (abstract)
Note	Definition of the IEEE1722Tp protocol. **Tags:** atp.Status=candidate
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Subclasses	IEEE1722TpAcfConnection, *IEEE1722TpAvConnection*

▽

△

Class	***IEEE1722TpConnection*** (abstract)			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Direction	Communication DirectionType	0..1	attr	Communication Direction of the IEEE1722TpConnection. **Tags:** atp.Status=candidate This Attribute is only used by the AUTOSAR Adaptive Platform.
destinationMac Address	MacAddressString	0..1	attr	Optional definition of the destination MAC address for this stream. If no given then macAddressStreamId is used as destination MAC address. **Tags:** atp.Status=candidate
globalTime Domain	GlobalTimeDomain	0..1	ref	Reference to the GlobalTimeDomain this IEEE1722Tp Connection shall be synchronized with. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeDomain.globalTimeDomain, globalTimeDomain.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
macAddress StreamId	MacAddressString	0..1	attr	MAC Address part of the Stream Id. **Tags:** atp.Status=candidate
uniqueStreamId	PositiveInteger	0..1	attr	Unique Id part of the Stream Id. **Tags:** atp.Status=candidate
version	PositiveInteger	0..1	attr	Version of the IEEE1722TP stream. **Tags:** atp.Status=candidate
vlanPriority	PositiveInteger	0..1	attr	Optional definition of the VLAN priority for this stream.

Table A.20: IEEE1722TpConnection

Class	**IPSecConfig**			
Note	IPsec is a protocol that is designed to provide "end-to-end" cryptographically-based security for IP network connections.			
Base	*ARObject*			
Aggregated by	NetworkEndpoint.ipSecConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipSecConfig Props	IPSecConfigProps	0..1	ref	Global IPsec configuration settings that are valid for all IPSecRules that are defined on the NetworkEndpoint.
ipSecRule	IPSecRule	*	aggr	IPSec rules and filters that are defined in the IPSecConfig for a specific NetworkEndpoint.

Table A.21: IPSecConfig

Class	**NetworkEndpoint**			
Note	The network endpoint defines the network addressing (e.g. IP-Address or MAC multicast address).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetPhysicalChannel.networkEndpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fullyQualified DomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
ipSecConfig	IPSecConfig	0..1	aggr	Optional IPSec configuration that provides security services for IP packets.

▽

△

Class	**NetworkEndpoint**			
network Endpoint Address	NetworkEndpoint Address	*	aggr	Definition of a Network Address. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=networkEndpointAddress.ipv4Address, networkEndpointAddress.ipv6Address, networkEndpoint Address.macMulticastGroup xml.namePlural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.

Table A.22: NetworkEndpoint

Class	***PortInterface*** (abstract)			
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*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSuspendToRamInterface*, *AbstractSynchronizedTimeBase Interface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallState SwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, Network ManagementPortInterface, *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 This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.23: PortInterface

Class	***PortPrototype*** (abstract)			
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.

▽

△

Class	***PortPrototype*** (abstract)			
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. This Attribute is only used by the AUTOSAR Adaptive Platform.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table A.24: PortPrototype

Class	**Process**			
Note	This meta-class provides information required to execute the referenced `Executable`. **Tags:** atp.recommendedPackage=Processes This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference.
executable	Executable	*	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef
functionCluster Affiliation	String	0..1	attr	This attribute specifies which functional cluster the Process is affiliated with.
numberOf RestartAttempts	PositiveInteger	0..1	attr	This attribute defines how often a process shall be restarted if the start fails. numberOfRestartAttempts = "0" OR Attribute not existing, start once numberOfRestartAttempts = "1", start a second time
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process. This attribute is used to support the modeling of execution dependencies that utilize the condition of process state. Please note that the process states may not be modeled arbitrarily at any stage of the AUTOSAR workflow because the supported states are standardized in the context of the SWS Execution Management [10].
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations.

Table A.25: Process

Class	RPortPrototype			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table A.26: RPortPrototype

Class	RawDataStreamClientInterface			
Note	This meta-class represents the necessary capabilities for raw data streaming on the client side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamClientInterface supports the following capabilities without further modeling: • connect: set up the communication channel • shutdown: close the communication channel • write: send data down the communication channel • read: access incoming data on the communication channel **Tags:** atp.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.27: RawDataStreamClientInterface

Class	RawDataStreamEthernetTcpUdpCredentials			
Note	This-meta-class represents the ability to create a configuration of network credentials for a raw data stream connection over TCP and UDP (inherited from base class). This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractRawDataStreamEthernetCredentials*, *Describable*			
Aggregated by	EthernetRawDataStreamRemoteServerConfig.unicastCredentials			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpPort	PositiveInteger	0..1	attr	This attribute represents the configuration of a TCP port number.

Table A.28: RawDataStreamEthernetTcpUdpCredentials

Class	RawDataStreamEthernetUdpCredentials			
Note	This-meta-class represents the ability to create a configuration of network credentials for a raw data stream connection over UDP. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARObject*, *AbstractRawDataStreamEthernetCredentials*, *Describable*			
Aggregated by	EthernetRawDataStreamRemoteClientConfig.multicastCredentials, EthernetRawDataStreamRemote ClientConfig.unicastUdpCredentials, EthernetRawDataStreamRemoteServerConfig.multicastCredentials			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.29: RawDataStreamEthernetUdpCredentials

Class	***RawDataStreamMapping*** (abstract)			
Note	This meta-class acts as an abstract base class for mapping raw data streams to the application software. This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*EthernetMacRawDataStreamMapping*, *EthernetRawDataStreamMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deployment	RawDataStream Deployment	0..1	ref	This reference identifies the applicable RawDataStream Deployment.
portPrototype	RPortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the raw data stream to the application. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced Port Prototype is executed.

Table A.30: RawDataStreamMapping

Class	**RawDataStreamServerInterface**			
Note	This meta-class represents the necessary capabilities for raw data streaming on the server side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamServerInterface supports the following capabilities without further modeling: • waitForConnection: wait until a communication channel is set up. • shutdown: close the communication channel • write: send data down the communication channel • read: access incoming data on the communication channel **Tags:** atp.recommendedPackage=RawDataStreamInterfaces This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.31: RawDataStreamServerInterface

Class	**TlsSecureComProps**			
Note	Configuration of the Transport Layer Security protocol (TLS). **Tags:** atp.recommendedPackage=SecureComProps This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared (i.e. applicable for each of the aggregated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase.

▽

△

Class	**TlsSecureComProps**			
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	Collection of supported cipher suites that are used to negotiate the security settings for a network connection defined by the ServiceInstanceToMachineMapping.

Table A.32: TlsSecureComProps

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 Platform Extension Interfaces (normative)

This functional cluster does not specify any Platform Extension Interfaces.

D Not implemented requirements

This functional cluster implements all functional requirements specified in the corresponding requirement specifications.

E History of Constraints and Specification Items

This chapter provides an overview of the history of constraints and specification items. Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

E.1 Constraint and Specification Item Changes between AUTOSAR Release R24-11 and R25-11

E.1.1 Added Specification Items in R25-11

none

E.1.2 Changed Specification Items in R25-11

Number	Heading
[SWS_RDS_10481]	Definition of API class ara::rds::RawDataStreamClient
[SWS_RDS_10482]	Definition of API function ara::rds::RawDataStreamClient::Create
[SWS_RDS_10484]	Definition of API function ara::rds::RawDataStreamClient::Connect
[SWS_RDS_10485]	Definition of API function ara::rds::RawDataStreamClient::Shutdown
[SWS_RDS_10486]	Definition of API function ara::rds::RawDataStreamClient::ReadData
[SWS_RDS_10487]	Definition of API function ara::rds::RawDataStreamClient::WriteData
[SWS_RDS_11291]	Definition of API class ara::rds::RdsException
[SWS_RDS_11292]	Definition of API function ara::rds::RdsException::RdsException
[SWS_RDS_11293]	Definition of API class ara::rds::RdsErrorDomain
[SWS_RDS_11294]	Definition of API function ara::rds::RdsErrorDomain::RdsErrorDomain
[SWS_RDS_11295]	Definition of API function ara::rds::RdsErrorDomain::Name
[SWS_RDS_11296]	Definition of API function ara::rds::RdsErrorDomain::Message
[SWS_RDS_11297]	Definition of API function ara::rds::RdsErrorDomain::ThrowAsException
[SWS_RDS_11298]	Definition of API function ara::rds::GetRdsErrorDomain
[SWS_RDS_11299]	Definition of API function ara::rds::MakeErrorCode
[SWS_RDS_11305]	Definition of API function ara::rds::RawDataStreamClient::RawDataStream Client
[SWS_RDS_11307]	Definition of API function ara::rds::RawDataStreamClient::Connect
[SWS_RDS_11309]	Definition of API function ara::rds::RawDataStreamClient::ReadData
[SWS_RDS_11310]	Definition of API function ara::rds::RawDataStreamClient::WriteData
[SWS_RDS_11311]	Definition of API class ara::rds::RawDataStreamServer
[SWS_RDS_11312]	Definition of API function ara::rds::RawDataStreamServer::Create

▽

△

Number	Heading
[SWS_RDS_11316]	Definition of API function ara::rds::RawDataStreamServer::RawDataStream Server
[SWS_RDS_11318]	Definition of API function ara::rds::RawDataStreamServer::WaitFor Connection
[SWS_RDS_11319]	Definition of API function ara::rds::RawDataStreamServer::WaitFor Connection
[SWS_RDS_11320]	Definition of API function ara::rds::RawDataStreamServer::Shutdown
[SWS_RDS_11322]	Definition of API function ara::rds::RawDataStreamServer::ReadData
[SWS_RDS_11323]	Definition of API function ara::rds::RawDataStreamServer::ReadData
[SWS_RDS_11324]	Definition of API function ara::rds::RawDataStreamServer::WriteData
[SWS_RDS_11325]	Definition of API function ara::rds::RawDataStreamServer::WriteData
[SWS_RDS_11326]	Definition of API type ara::rds::RdsErrorDomain::Errc
[SWS_RDS_11327]	Definition of API type ara::rds::RdsErrorDomain::Exception
[SWS_RDS_12367]	Definition of API enum ara::rds::RdsErrc
[SWS_RDS_90311]	Definition of API class ara::rds::IEEE1722RawDataStreamConsumer
[SWS_RDS_90312]	Definition of API function ara::rds::IEEE1722RawDataStream Consumer::Create
[SWS_RDS_90319]	Definition of API function ara::rds::IEEE1722RawDataStream Consumer::Connect
[SWS_RDS_90320]	Definition of API function ara::rds::IEEE1722RawDataStream Consumer::Shutdown
[SWS_RDS_90321]	Definition of API function ara::rds::IEEE1722RawDataStream Consumer::ReadData
[SWS_RDS_90322]	Definition of API class ara::rds::IEEE1722RawDataStreamProducer
[SWS_RDS_90323]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::Create
[SWS_RDS_90330]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::Connect
[SWS_RDS_90331]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::Shutdown
[SWS_RDS_90332]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::WriteData
[SWS_RDS_99006]	Timeout handling

Table E.1: Changed Specification Items in R25-11

E.1.3 Deleted Specification Items in R25-11

none

E.1.4 Added Constraints in R25-11

none

E.1.5 Changed Constraints in R25-11

none

E.1.6 Deleted Constraints in R25-11

none

E.2 Constraint and Specification Item Changes between AUTOSAR Release R23-11 and R24-11

E.2.1 Added Specification Items in R24-11

Number	Heading
[SWS_RDS_10511]	Creation of RawDataStreamClient instance
[SWS_RDS_10512]	Creation of RawDataStreamServer instance
[SWS_RDS_10513]	Setup RawDataStreamClient connection
[SWS_RDS_10514]	Connect RawDataStreamServer to incoming connection
[SWS_RDS_10515]	Shutdown RawDataStreamClient connection
[SWS_RDS_10516]	Shutdown RawDataStreamServer connection
[SWS_RDS_10517]	Read data from a RawDataStreamClient connection
[SWS_RDS_10518]	Read data from a RawDataStreamServer connection
[SWS_RDS_10519]	Write data to a RawDataStreamClient connection
[SWS_RDS_10520]	Write data to a RawDataStreamServer connection
[SWS_RDS_10525]	Supported types to create an instance of `IEEE1722RawDataStreamConsumer` or `IEEE1722RawDataStreamProducer`
[SWS_RDS_10526]	Prepare a local connection to an `IEEE1722RawDataStream`
[SWS_RDS_10527]	Data link layer configuration of an `IEEE1722RawDataStreamConsumer` using IEEE1722 protocol
[SWS_RDS_10528]	Data link layer communication configuration of an `IEEE1722RawDataStreamConsumer` using IEEE1722 protocol
[SWS_RDS_10529]	Data link layer configuration of an `IEEE1722RawDataStreamProducer` using IEEE1722 protocol
[SWS_RDS_10530]	Data link layer communication configuration of an `IEEE1722RawDataStreamProducer` using IEEE1722 protocol
[SWS_RDS_10531]	Socket Options configuration
[SWS_RDS_10532]	Derive IEEE1722 protocol configuration at creation of an `IEEE1722RawDataStreamConsumer` instance
[SWS_RDS_10533]	Derive IEEE1722 protocol configuration at creation of an `IEEE1722RawDataStreamProducer` instance

▽

△

Number	Heading
[SWS_RDS_10534]	Error handling if IEEE1722 protocol configuration is derived
[SWS_RDS_10535]	Establish local communication connection to an IEEE1722RawDataStream
[SWS_RDS_10536]	Error handling for establishing a local communication connection to an IEEE1722RawDataStream
[SWS_RDS_10537]	Shutdown local communication connection to an IEEE1722RawDataStream
[SWS_RDS_10538]	Error handling for shutdown a local communication connection to an IEEE1722RawDataStream
[SWS_RDS_10539]	Destructor behavior for local communication connection to an IEEE1722RawDataStream
[SWS_RDS_10540]	Restrictions on using IEEE1722RawDataStream
[SWS_RDS_10541]	Read data from an IEEE1722RawDataStream
[SWS_RDS_10542]	Inspection of AVTPDU-common-header fields
[SWS_RDS_10543]	Inspection of AVTPDU-common-stream-header fields
[SWS_RDS_10544]	Consistency checks for stream header field values of AVTP common-stream-header format, AVTP stream data subtype CRF and NTSCF
[SWS_RDS_10545]	Write data to an IEEE1722RawDataStream
[SWS_RDS_10546]	Preparation of IEEE1722 header field value "sequence number"
[SWS_RDS_10547]	Preparation of IEEE1722 header field value "stream id"
[SWS_RDS_10548]	Determination of IEEE1722 header field value "time uncertain"
[SWS_RDS_10549]	Handling if length of AVTPDU-header and AVTP-payload exceeds maxiumn transmission unit
[SWS_RDS_10550]	Determination of IEEE1722 header field value "AVTP stream data subtype"
[SWS_RDS_10551]	Setting of IEEE1722 header field value "version"
[SWS_RDS_10552]	Determination of IEEE1722 header field value "media clock restart"
[SWS_RDS_10553]	Setting of IEEE1722 header field value "stream id valid"
[SWS_RDS_10554]	Determination of IEEE1722 header field value "sequence number"
[SWS_RDS_10555]	Setting of IEEE1722 header field value "timestamp uncertain"
[SWS_RDS_10556]	Creation of IEEE1722 header field value "stream id"
[SWS_RDS_10557]	Determination of IEEE1722 header field value "avtp timestamp"
[SWS_RDS_10558]	Setting of IEEE1722 header field value "stream data length"
[SWS_RDS_10559]	Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field is set to 0
[SWS_RDS_10560]	Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field is set to 1
[SWS_RDS_10561]	Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field is set to 1 and configured SPH field is set to 0
[SWS_RDS_10562]	Setting of IEEE1722 subtype stream 61883_IIDC header field values if configured tag field and configured SPH field are set to 1
[SWS_RDS_10563]	Setting of IEEE1722 subtype stream AAF header field values

▽

△

Number	Heading
[SWS_RDS_10564]	Setting of IEEE1722 subtype stream AAF header field values if AAF AVTP format PCM is indicated
[SWS_RDS_10565]	Setting of IEEE1722 subtype stream AAF header field values if AAF AVTP format AES3 is indicated
[SWS_RDS_10566]	Setting of IEEE1722 subtype stream NTSCF header field values
[SWS_RDS_10567]	Setting of IEEE1722 subtype stream TSCF header field values
[SWS_RDS_10568]	Setting of IEEE1722 subtype stream CRF header field values
[SWS_RDS_10569]	Setting of IEEE1722 subtype stream RVF header field values
[SWS_RDS_10570]	Error handling for read and write data to an disconnected `IEEE1722RawDataStream`
[SWS_RDS_10571]	Error handling for read and write data processing with system interrupt
[SWS_RDS_11326]	Definition of API type ara::rds::RawErrorDomain::Errc
[SWS_RDS_11327]	Definition of API type ara::rds::RawErrorDomain::Exception
[SWS_RDS_90225]	Definition of API class ara::rds::IEEE1722Datagram
[SWS_RDS_90226]	Definition of API variable ara::rds::IEEE1722Datagram::data
[SWS_RDS_90227]	Definition of API variable ara::rds::IEEE1722Datagram::stream_data_length
[SWS_RDS_90228]	Definition of API function ara::rds::IEEE1722Datagram::~IEEE1722Datagram
[SWS_RDS_90229]	Definition of API class ara::rds::IEEE1722DatagramAVTPDUCommon HeaderFields
[SWS_RDS_90230]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon HeaderFields::avtpStreamDataSubtype
[SWS_RDS_90231]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon HeaderFields::headerSpecific
[SWS_RDS_90232]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon HeaderFields::version
[SWS_RDS_90233]	Definition of API function ara::rds::IEEE1722DatagramAVTPDUCommon HeaderFields::~IEEE1722DatagramAVTPDUCommonHeaderFields
[SWS_RDS_90234]	Definition of API class ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields
[SWS_RDS_90235]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::mac_address
[SWS_RDS_90236]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::unique_id
[SWS_RDS_90237]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::avtp_timestamp
[SWS_RDS_90238]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::mr
[SWS_RDS_90239]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::tv
[SWS_RDS_90240]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::sequenceNum

▽

△

Number	Heading
[SWS_RDS_90241]	Definition of API variable ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::tu
[SWS_RDS_90242]	Definition of API function ara::rds::IEEE1722DatagramAVTPDUCommon StreamHeaderFields::~IEEE1722DatagramAVTPDUCommonStreamHeader Fields
[SWS_RDS_90243]	Definition of API class ara::rds::IEEE1722Datagram61883_IIDC
[SWS_RDS_90244]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::tag
[SWS_RDS_90245]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::channel
[SWS_RDS_90246]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::tcode
[SWS_RDS_90247]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::sy
[SWS_RDS_90248]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::qi_1
[SWS_RDS_90249]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::sid
[SWS_RDS_90250]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::dbs
[SWS_RDS_90251]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fn
[SWS_RDS_90252]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::qpc
[SWS_RDS_90253]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::sph
[SWS_RDS_90254]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::dbc
[SWS_RDS_90255]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::qi_2
[SWS_RDS_90256]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fmt
[SWS_RDS_90257]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fdf_ without_sph
[SWS_RDS_90258]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::syt
[SWS_RDS_90259]	Definition of API variable ara::rds::IEEE1722Datagram61883_IIDC::fdf_ with_sph
[SWS_RDS_90260]	Definition of API function ara::rds::IEEE1722Datagram61883_ IIDC::~IEEE1722Datagram61883_IIDC
[SWS_RDS_90261]	Definition of API class ara::rds::IEEE1722DatagramAAF
[SWS_RDS_90262]	Definition of API variable ara::rds::IEEE1722DatagramAAF::format
[SWS_RDS_90263]	Definition of API variable ara::rds::IEEE1722DatagramAAF::sp
[SWS_RDS_90264]	Definition of API variable ara::rds::IEEE1722DatagramAAF::evt
[SWS_RDS_90265]	Definition of API variable ara::rds::IEEE1722DatagramAAF::nsr
[SWS_RDS_90266]	Definition of API variable ara::rds::IEEE1722DatagramAAF::channels_per_ frame
[SWS_RDS_90267]	Definition of API variable ara::rds::IEEE1722DatagramAAF::bit_depth
[SWS_RDS_90268]	Definition of API variable ara::rds::IEEE1722DatagramAAF::nfr
[SWS_RDS_90269]	Definition of API variable ara::rds::IEEE1722DatagramAAF::streams_per_ frame
[SWS_RDS_90270]	Definition of API variable ara::rds::IEEE1722DatagramAAF::aes3_data_ type_h
[SWS_RDS_90271]	Definition of API variable ara::rds::IEEE1722DatagramAAF::aes3_dt_ref

▽

△

Number	Heading
[SWS_RDS_90272]	Definition of API variable ara::rds::IEEE1722DatagramAAF::aes3_data_type_l
[SWS_RDS_90273]	Definition of API function ara::rds::IEEE1722Datagram AAF::~IEEE1722DatagramAAF
[SWS_RDS_90274]	Definition of API class ara::rds::IEEE1722DatagramTSCF
[SWS_RDS_90275]	Definition of API function ara::rds::IEEE1722Datagram TSCF::~IEEE1722DatagramTSCF
[SWS_RDS_90276]	Definition of API class ara::rds::IEEE1722DatagramRVF
[SWS_RDS_90277]	Definition of API variable ara::rds::IEEE1722DatagramRVF::active_pixels
[SWS_RDS_90278]	Definition of API variable ara::rds::IEEE1722DatagramRVF::total_lines
[SWS_RDS_90279]	Definition of API variable ara::rds::IEEE1722DatagramRVF::ap
[SWS_RDS_90280]	Definition of API variable ara::rds::IEEE1722DatagramRVF::f
[SWS_RDS_90281]	Definition of API variable ara::rds::IEEE1722DatagramRVF::ef
[SWS_RDS_90282]	Definition of API variable ara::rds::IEEE1722DatagramRVF::evt
[SWS_RDS_90283]	Definition of API variable ara::rds::IEEE1722DatagramRVF::pd
[SWS_RDS_90284]	Definition of API variable ara::rds::IEEE1722DatagramRVF::i
[SWS_RDS_90285]	Definition of API variable ara::rds::IEEE1722DatagramRVF::pixel_depth
[SWS_RDS_90286]	Definition of API variable ara::rds::IEEE1722DatagramRVF::pixel_format
[SWS_RDS_90287]	Definition of API variable ara::rds::IEEE1722DatagramRVF::frame_rate
[SWS_RDS_90288]	Definition of API variable ara::rds::IEEE1722DatagramRVF::colorspace
[SWS_RDS_90289]	Definition of API variable ara::rds::IEEE1722DatagramRVF::num_lines
[SWS_RDS_90290]	Definition of API variable ara::rds::IEEE1722DatagramRVF::i_seq_num
[SWS_RDS_90291]	Definition of API variable ara::rds::IEEE1722DatagramRVF::line_number
[SWS_RDS_90292]	Definition of API function ara::rds::IEEE1722Datagram RVF::~IEEE1722DatagramRVF
[SWS_RDS_90293]	Definition of API class ara::rds::IEEE1722DatagramCRF
[SWS_RDS_90294]	Definition of API variable ara::rds::IEEE1722DatagramCRF::mr
[SWS_RDS_90295]	Definition of API variable ara::rds::IEEE1722DatagramCRF::fs
[SWS_RDS_90296]	Definition of API variable ara::rds::IEEE1722DatagramCRF::tu
[SWS_RDS_90297]	Definition of API variable ara::rds::IEEE1722DatagramCRF::sequenceNum
[SWS_RDS_90298]	Definition of API variable ara::rds::IEEE1722DatagramCRF::type
[SWS_RDS_90299]	Definition of API variable ara::rds::IEEE1722DatagramCRF::mac_address
[SWS_RDS_90300]	Definition of API variable ara::rds::IEEE1722DatagramCRF::unique_id
[SWS_RDS_90301]	Definition of API variable ara::rds::IEEE1722DatagramCRF::pull
[SWS_RDS_90302]	Definition of API variable ara::rds::IEEE1722DatagramCRF::base_frequency
[SWS_RDS_90303]	Definition of API variable ara::rds::IEEE1722DatagramCRF::timestamp_interval
[SWS_RDS_90304]	Definition of API function ara::rds::IEEE1722Datagram CRF::~IEEE1722DatagramCRF

▽

△

Number	Heading
[SWS_RDS_90305]	Definition of API class ara::rds::IEEE1722DatagramNTSCF
[SWS_RDS_90306]	Definition of API variable ara::rds::IEEE1722DatagramNTSCF::ntscf_data_length
[SWS_RDS_90307]	Definition of API variable ara::rds::IEEE1722DatagramNTSCF::sequenceNum
[SWS_RDS_90308]	Definition of API variable ara::rds::IEEE1722DatagramNTSCF::mac_address
[SWS_RDS_90309]	Definition of API variable ara::rds::IEEE1722DatagramNTSCF::unique_id
[SWS_RDS_90310]	Definition of API function ara::rds::IEEE1722DatagramNTSCF::~IEEE1722DatagramNTSCF
[SWS_RDS_90311]	Definition of API class ara::rds::IEEE1722RawDataStreamConsumer
[SWS_RDS_90312]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::Create
[SWS_RDS_90313]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::IEEE1722RawDataStreamConsumer
[SWS_RDS_90314]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::IEEE1722RawDataStreamConsumer
[SWS_RDS_90315]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::operator=
[SWS_RDS_90316]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::IEEE1722RawDataStreamConsumer
[SWS_RDS_90317]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::operator=
[SWS_RDS_90318]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::~IEEE1722RawDataStreamConsumer
[SWS_RDS_90319]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::Connect
[SWS_RDS_90320]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::Shutdown
[SWS_RDS_90321]	Definition of API function ara::rds::IEEE1722RawDataStreamConsumer::ReadData
[SWS_RDS_90322]	Definition of API class ara::rds::IEEE1722RawDataStreamProducer
[SWS_RDS_90323]	Definition of API function ara::rds::IEEE1722RawDataStreamProducer::Create
[SWS_RDS_90324]	Definition of API function ara::rds::IEEE1722RawDataStreamProducer::IEEE1722RawDataStreamProducer
[SWS_RDS_90325]	Definition of API function ara::rds::IEEE1722RawDataStreamProducer::IEEE1722RawDataStreamProducer
[SWS_RDS_90326]	Definition of API function ara::rds::IEEE1722RawDataStreamProducer::operator=
[SWS_RDS_90327]	Definition of API function ara::rds::IEEE1722RawDataStreamProducer::IEEE1722RawDataStreamProducer
[SWS_RDS_90328]	Definition of API function ara::rds::IEEE1722RawDataStreamProducer::operator=

▽

△

Number	Heading
[SWS_RDS_90329]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::~IEEE1722RawDataStreamProducer
[SWS_RDS_90330]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::Connect
[SWS_RDS_90331]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::Shutdown
[SWS_RDS_90332]	Definition of API function ara::rds::IEEE1722RawDataStream Producer::WriteData

Table E.2: Added Specification Items in R24-11

E.2.2 Changed Specification Items in R24-11

Number	Heading
[SWS_RDS_10482]	Definition of API function ara::rds::RawDataStreamClient::Create
[SWS_RDS_10483]	Definition of API function ara::rds::RawDataStreamClient::~RawDataStream Client
[SWS_RDS_10484]	Definition of API function ara::rds::RawDataStreamClient::Connect
[SWS_RDS_10485]	Definition of API function ara::rds::RawDataStreamClient::Shutdown
[SWS_RDS_10486]	Definition of API function ara::rds::RawDataStreamClient::ReadData
[SWS_RDS_10487]	Definition of API function ara::rds::RawDataStreamClient::WriteData
[SWS_RDS_11292]	Definition of API function ara::rds::RawException::RawException
[SWS_RDS_11294]	Definition of API function ara::rds::RawErrorDomain::RawErrorDomain
[SWS_RDS_11295]	Definition of API function ara::rds::RawErrorDomain::Name
[SWS_RDS_11296]	Definition of API function ara::rds::RawErrorDomain::Message
[SWS_RDS_11297]	Definition of API function ara::rds::RawErrorDomain::ThrowAsException
[SWS_RDS_11298]	Definition of API function ara::rds::GetRawErrorDomain
[SWS_RDS_11299]	Definition of API function ara::rds::MakeErrorCode
[SWS_RDS_11303]	Definition of API function ara::rds::RawDataStreamClient::RawDataStream Client
[SWS_RDS_11304]	Definition of API function ara::rds::RawDataStreamClient::operator=
[SWS_RDS_11305]	Definition of API function ara::rds::RawDataStreamClient::RawDataStream Client
[SWS_RDS_11306]	Definition of API function ara::rds::RawDataStreamClient::operator=
[SWS_RDS_11307]	Definition of API function ara::rds::RawDataStreamClient::Connect
[SWS_RDS_11309]	Definition of API function ara::rds::RawDataStreamClient::ReadData
[SWS_RDS_11310]	Definition of API function ara::rds::RawDataStreamClient::WriteData
[SWS_RDS_11312]	Definition of API function ara::rds::RawDataStreamServer::Create
[SWS_RDS_11313]	Definition of API function ara::rds::RawDataStreamServer::~RawData StreamServer

▽

△

Number	Heading
[SWS_RDS_11314]	Definition of API function ara::rds::RawDataStreamServer::RawDataStream Server
[SWS_RDS_11315]	Definition of API function ara::rds::RawDataStreamServer::operator=
[SWS_RDS_11316]	Definition of API function ara::rds::RawDataStreamServer::RawDataStream Server
[SWS_RDS_11317]	Definition of API function ara::rds::RawDataStreamServer::operator=
[SWS_RDS_11318]	Definition of API function ara::rds::RawDataStreamServer::WaitFor Connection
[SWS_RDS_11319]	Definition of API function ara::rds::RawDataStreamServer::WaitFor Connection
[SWS_RDS_11320]	Definition of API function ara::rds::RawDataStreamServer::Shutdown
[SWS_RDS_11322]	Definition of API function ara::rds::RawDataStreamServer::ReadData
[SWS_RDS_11323]	Definition of API function ara::rds::RawDataStreamServer::ReadData
[SWS_RDS_11324]	Definition of API function ara::rds::RawDataStreamServer::WriteData
[SWS_RDS_11325]	Definition of API function ara::rds::RawDataStreamServer::WriteData
[SWS_RDS_12367]	Definition of API enum ara::rds::RawErrc
[SWS_RDS_90007]	Restrictions on using RawDataStreams

Table E.3: Changed Specification Items in R24-11

E.2.3 Deleted Specification Items in R24-11

Number	Heading
[SWS_RDS_10488]	Raw data stream header file existence
[SWS_RDS_10489]	Raw data stream header file namespace
[SWS_RDS_10490]	Data Type declarations in Raw data stream header file
[SWS_RDS_99025]	Raw errors domain

Table E.4: Deleted Specification Items in R24-11

E.2.4 Added Constraints in R24-11

Number	Heading
[SWS_RDS_CONSTR_00001]	Configurable Namespace for RawDataStream

Table E.5: Added Constraints in R24-11

E.2.5 Changed Constraints in R24-11

none

E.2.6 Deleted Constraints in R24-11

none

E.2.7 Added Specification Items in R23-11

Number	Heading
[SWS_RDS_10476]	Defining a RawDataStream
[SWS_RDS_10477]	Connect stream link
[SWS_RDS_10478]	Shutdown stream link
[SWS_RDS_10479]	Read data from stream
[SWS_RDS_10480]	Write data to stream
[SWS_RDS_10481]	Definition of API class ara::rds::RawDataStreamClient
[SWS_RDS_10482]	Definition of API function ara::rds::RawDataStreamClient::Create
[SWS_RDS_10483]	Definition of API function ara::rds::RawDataStreamClient::~RawDataStream Client
[SWS_RDS_10484]	Definition of API function ara::rds::RawDataStreamClient::Connect
[SWS_RDS_10485]	Definition of API function ara::rds::RawDataStreamClient::Shutdown
[SWS_RDS_10486]	Definition of API function ara::rds::RawDataStreamClient::ReadData
[SWS_RDS_10487]	Definition of API function ara::rds::RawDataStreamClient::WriteData
[SWS_RDS_10488]	Raw data stream header file existence
[SWS_RDS_10489]	Raw data stream header file namespace
[SWS_RDS_10490]	Data Type declarations in Raw data stream header file
[SWS_RDS_10508]	Destructor behavior when Shutdown stream link
[SWS_RDS_11291]	Definition of API class ara::rds::RawException
[SWS_RDS_11292]	Definition of API function ara::rds::RawException::RawException
[SWS_RDS_11293]	Definition of API class ara::rds::RawErrorDomain
[SWS_RDS_11294]	Definition of API function ara::rds::RawErrorDomain::RawErrorDomain
[SWS_RDS_11295]	Definition of API function ara::rds::RawErrorDomain::Name
[SWS_RDS_11296]	Definition of API function ara::rds::RawErrorDomain::Message
[SWS_RDS_11297]	Definition of API function ara::rds::RawErrorDomain::ThrowAsException
[SWS_RDS_11298]	Definition of API function ara::rds::GetRawErrorDomain
[SWS_RDS_11299]	Definition of API function ara::rds::MakeErrorCode
[SWS_RDS_11300]	Definition of API class ara::rds::ReadDataResult
[SWS_RDS_11301]	Definition of API variable ara::rds::ReadDataResult::data

▽

△

Number	Heading
[SWS_RDS_11302]	Definition of API variable ara::rds::ReadDataResult::numberOfBytes
[SWS_RDS_11303]	Definition of API function ara::rds::RawDataStreamClient::RawDataStream Client
[SWS_RDS_11304]	Definition of API function ara::rds::RawDataStreamClient::operator=
[SWS_RDS_11305]	Definition of API function ara::rds::RawDataStreamClient::RawDataStream Client
[SWS_RDS_11306]	Definition of API function ara::rds::RawDataStreamClient::operator=
[SWS_RDS_11307]	Definition of API function ara::rds::RawDataStreamClient::Connect
[SWS_RDS_11309]	Definition of API function ara::rds::RawDataStreamClient::ReadData
[SWS_RDS_11310]	Definition of API function ara::rds::RawDataStreamClient::WriteData
[SWS_RDS_11311]	Definition of API class ara::rds::RawDataStreamServer
[SWS_RDS_11312]	Definition of API function ara::rds::RawDataStreamServer::Create
[SWS_RDS_11313]	Definition of API function ara::rds::RawDataStreamServer::~RawData StreamServer
[SWS_RDS_11314]	Definition of API function ara::rds::RawDataStreamServer::RawDataStream Server
[SWS_RDS_11315]	Definition of API function ara::rds::RawDataStreamServer::operator=
[SWS_RDS_11316]	Definition of API function ara::rds::RawDataStreamServer::RawDataStream Server
[SWS_RDS_11317]	Definition of API function ara::rds::RawDataStreamServer::operator=
[SWS_RDS_11318]	Definition of API function ara::rds::RawDataStreamServer::WaitFor Connection
[SWS_RDS_11319]	Definition of API function ara::rds::RawDataStreamServer::WaitFor Connection
[SWS_RDS_11320]	Definition of API function ara::rds::RawDataStreamServer::Shutdown
[SWS_RDS_11322]	Definition of API function ara::rds::RawDataStreamServer::ReadData
[SWS_RDS_11323]	Definition of API function ara::rds::RawDataStreamServer::ReadData
[SWS_RDS_11324]	Definition of API function ara::rds::RawDataStreamServer::WriteData
[SWS_RDS_11325]	Definition of API function ara::rds::RawDataStreamServer::WriteData
[SWS_RDS_12367]	Definition of API enum ara::rds::RawErrc
[SWS_RDS_90007]	Restrictions on using RawDataStreams
[SWS_RDS_90211]	Secure UDP and TCP channel creation for TLS and DTLS
[SWS_RDS_90212]	Using secure TLS, DTLS channels
[SWS_RDS_90213]	TLS secure channel for raw data streams using reliable transport
[SWS_RDS_90214]	DTLS secure channel for methods using unreliable transport
[SWS_RDS_90215]	IPsec secure channel between communication nodes and Transport of Raw Data Stream communication over an IPsec security association
[SWS_RDS_90216]	Socket Options configuration
[SWS_RDS_90217]	TLS properties configuration
[SWS_RDS_99004]	Ethernet endpoint configuration

▽

△

Number	Heading
[SWS_RDS_99005]	Wait for incoming connections
[SWS_RDS_99006]	Timeout handling
[SWS_RDS_99025]	Raw errors domain

Table E.6: Added Specification Items in R23-11

E.2.8 Changed Specification Items in R23-11

none

E.2.9 Deleted Specification Items in R23-11

none