Document Title	Specification of Data Distribution Service for Classic Platform
Document Category	Software specification (SWS)
Document UID	1069
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Added DMA copy support
2025-11-27	R25-11	AUTOSAR Release Management	• Added Service Oriented Architecture management • Added SPDP, SEDP, Service Discovery engines management • ECU model extension to handle full SOA and Discovery features • Added Appendix explanation chapters • Moved architecture from LdCom to Com
2024-11-27	R24-11	AUTOSAR Release Management	• Refinement of some specification item • Added details on reception and transmission requirements • Clarification regarding remote participants • Validation of all specification items
2023-11-23	R23-11	AUTOSAR Release Management	• ECU configuration model refactoring • Added details on Tx and Rx path and queue management • API renaming and clarification • Improved description of safety requirements and management
2022-11-24	R22-11	AUTOSAR Release Management	• Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 8
1.1 DDS protocol overview . . . 8
2 Acronyms and Abbreviations 9
2.1 Acronyms . . . 9
2.2 Abbreviations . . . 9
2.3 Glossary . . . 9
3 Related documentation 10
3.1 Input documents & related standards and norms . . . 10
3.2 Related specification . . . 11
4 Constraints and assumptions 12
4.1 Constraints and assumptions . . . 12
4.1.1 Assumptions . . . 12
4.1.2 Limitations . . . 12
4.1.3 Constraints . . . 13
4.2 Applicability to car domains . . . 14
5 Dependencies to other modules 15
5.1 RTE (BSW Scheduler) . . . 15
5.2 PDU Router . . . 15
5.3 StbM . . . 15
5.4 Default Error Tracer . . . 15
5.5 Crypto Service Manager . . . 15
5.6 Cyclic Redundancy Check . . . 16
5.7 Socket Adaptor . . . 16
6 Requirements Tracing 17
7 Functional specification 21
7.1 Overview . . . 21
7.1.1 Communication paths . . . 22
7.1.2 Service Oriented Architecture . . . 24
7.1.2.1 Service interface elements with Dds . . . 25
7.1.2.2 Bidirectional service interface elements . . . 27
7.1.3 QoS Management . . . 28
7.1.3.1 TRANSPORT_PRIORITY QoS mapping . . . 28
7.1.4 Security Mechanisms . . . 29
7.1.5 Safety Mechanisms . . . 30
7.1.6 Discovery mechanism . . . 30
7.1.6.1 Socket connections control . . . 33
7.2 General Requirements . . . 35
7.2.1 Communication requirements . . . 35

7.2.1.1 Serialization requirements . . . 35
7.2.1.2 Deserialization requirements . . . 37
7.2.1.3 Transmission Queue management . . . 37
7.2.1.4 Transmission requirements . . . 38
7.2.1.5 Reception Queue management . . . 40
7.2.1.6 Reception requirements . . . 41
7.2.1.7 Timing requirements . . . 42
7.2.1.8 Service Oriented Architecture requirements . . . 42
7.2.1.9 Dynamic discovery requirements . . . 50
7.2.1.10 Asynchronous copying to local buffer . . . 52
7.2.2 Security requirements . . . 54
7.2.3 Safety requirements . . . 56
7.3 Error Classification . . . 57
7.3.1 Development Errors . . . 57
7.3.2 Runtime Errors . . . 57
7.3.3 Production Errors . . . 58
8 API specification 59
8.1 Imported types . . . 59
8.2 Type definitions . . . 59
8.2.1 Dds_ConfigType . . . 59
8.3 Function definitions . . . 60
8.3.1 Start-up and control services . . . 60
8.3.1.1 Dds_Init . . . 60
8.3.1.2 Dds_ReleaseRxBuffer . . . 61
8.3.2 Communication services . . . 61
8.3.2.1 Dds_Transmit . . . 61
8.3.2.2 Dds_RxIndication . . . 62
8.3.2.3 Dds_TxConfirmation . . . 63
8.3.2.4 Dds_TriggerTransmit . . . 64
8.4 Scheduled functions . . . 66
8.4.1 Dds_MainFunction_Rx . . . 66
8.4.2 Dds_MainFunction_Tx . . . 66
8.5 Expected interfaces . . . 67
8.5.1 Mandatory interfaces . . . 68
8.5.2 Optional interfaces . . . 68
8.5.3 Configurable interfaces . . . 69
9 Sequence diagrams 70
9.1 Transmission . . . 70
9.1.1 Dds message transmission . . . 70
9.1.2 Dds message transmission confirmation . . . 70
9.1.3 Dds message trigger transmission . . . 71
9.2 Reception . . . 71

9.2.1 Dds received indication event . . . 71
9.3 Discovery . . . 72
9.3.1 Remote endpoint address setting . . . 72
9.3.2 Setting SoAdConnection to unused . . . 73
10 Configuration specification 74
10.1 How to read this chapter . . . 74
10.2 Containers and configuration parameters . . . 74
10.2.1 Dds . . . 74
10.2.2 Dds General . . . 75
10.2.3 Dds Config . . . 77
10.2.3.1 DdsAppDataPduCollection . . . 78
10.2.3.2 DdsQueueCollection . . . 82
10.2.3.3 DdsDomainParticipantCollection . . . 89
10.2.3.4 DdsDomainMulticastRtpsCollection . . . 140
10.2.3.5 DdsDomainDpMulticastRtpsPdusCollection . . . 142
10.2.3.6 DdsQoSPolicies . . . 145
10.3 Published Information . . . 168
A Not applicable requirements 169
B Change history of AUTOSAR traceable items 170
B.1 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 170
B.1.1 Added Specification Items in R25-11 . . . 170
B.1.2 Changed Specification Items in R25-11 . . . 173
B.1.3 Deleted Specification Items in R25-11 . . . 173
B.1.4 Added Constraints in R25-11 . . . 173
B.1.5 Changed Constraints in R25-11 . . . 175
B.1.6 Deleted Constraints in R25-11 . . . 175
B.2 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 175
B.2.1 Added Specification Items in R26-11 . . . 175
B.2.2 Changed Specification Items in R26-11 . . . 176
B.2.3 Deleted Specification Items in R26-11 . . . 176
B.2.4 Added Constraints in R26-11 . . . 176
B.2.5 Changed Constraints in R26-11 . . . 176
B.2.6 Deleted Constraints in R26-11 . . . 176
C Referenced Meta Classes 177
D Detailed description of Service Interface elements 187
D.1 Events . . . 187
D.2 Methods . . . 187
D.2.1 Request and Response Methods . . . 188
D.2.2 Fire and Forget Method with Arguments . . . 188

D.2.3 Fire and Forget Method without arguments 189
D.3 Fields 190
E Bidirectional service interface element functional communication scheme 192
F Examples 194
F.1 Transport layer configuration 194
F.1.1 Discovery connections configuration 196
F.2 SOME/IP DDS mixed architecture 197

Known Limitations

None.

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the **Data Distribution Service AUTOSAR Basic Software module (Dds)** .

1.1 DDS protocol overview

The **Data Distribution Service (DDS)** is a middleware protocol and API standard from the Object Management Group (OMG).

A preliminary overview of DDS can be found into chapter 4 of **AUTOSAR_FO_RS_DataDistributionService** [1] or referring directly the OMG standard [2].

2 Acronyms and Abbreviations

2.1 Acronyms

For acronyms and abbreviations that are not present in the table below, refer to [3, AUTOSAR glossary].

Acronym	Description
DdsXf	DDS Transformer
DW	Data Writer
DR	Data Reader
FSM	Finite State Machine

Table 2.1: Acronyms

2.2 Abbreviations

None.

2.3 Glossary

- **DDS/non-DDS PDU:** PDU containing/NOT containing DDS data (e.g., PDU listed/not listed into `DdsAppDataTxPduCollection`, `DdsAppDataRxPduCollection`, `DdsDomainParticipantUnicastRtpsPduCollection`, `DdsDomainMulticastRtpsPduCollection`, `DdsDomainParticipantDpUnicastRtpsPduCollection`, `DdsDomainDpMulticastRtpsPduCollection` containers).
- **Service interface element:** An atomic element belonging to a Service interface as defined in Chapter 7.1.2.1.
- **Unused SoAdConnection:** A `SoAdConnection` referencing a `SoAdSocketRemoteAddress` set according to [SWS_SoAd_00676]. In this specification the statements "reset a `SoAdConnection`" and "set to **unused** a `SoAdConnection`" have the same meaning.
- **Sample:** In the DDS protocol, a sample is the fundamental unit of data communication. It comprises both the data payload and the associated metadata including timestamps and identifiers which are essential for ensuring accurate interpretation, proper synchronization, and timely delivery between publishers and subscribers. In the AUTOSAR Classic context, an instance of a `VariableDataPrototype` or an `ClientServerOperation` constitutes the payload of the aforementioned sample.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Requirements on Data Distribution Service
AUTOSAR_FO_RS_DataDistributionService

[2] Data Distribution Service (DDS), Version 1.4
http://www.omg.org/spec/DDS/1.4

[3] Glossary
AUTOSAR_FO_TR_Glossary

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] Specification of Socket Adaptor
AUTOSAR_CP_SWS_SocketAdaptor

[6] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[7] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter

[8] DDS Interoperability Wire Protocol, Version 2.2
http://www.omg.org/spec/DDSI-RTPS/2.2

[9] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[10] Specification of Crypto Service Manager
AUTOSAR_CP_SWS_CryptoServiceManager

[11] Specification of CRC Library
AUTOSAR_CP_SWS_CRCLibrary

[12] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[13] System Template
AUTOSAR_CP_TPS_SystemTemplate

[14] RPC over DDS, Version 1.0
https://www.omg.org/spec/DDS-RPC/1.0

[15] Specification of DDS Service Communication Protocol
AUTOSAR_FO_PRS_DDSCommunicationProtocol

[16] DDS Security, Version 1.1
https://www.omg.org/spec/DDS-SECURITY/1.1

[17] ISO 26262:2018 Road vehicles -– Functional Safety

https://www.iso.org

[18] Specification of DDS Service Discovery Protocol
AUTOSAR_FO_PRS_DDSServiceDiscoveryProtocol

[19] Extensible and Dynamic Topic Types for DDS, Version 1.2
https://www.omg.org/spec/DDS-XTypes/1.2

[20] Specification of Platform Types for Classic Platform
AUTOSAR_CP_SWS_PlatformTypes

[21] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4], which is also valid for the Dds.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Dds.

4 Constraints and assumptions

4.1 Constraints and assumptions

4.1.1 Assumptions

- **DDS/non-DDS PDU concatenation.** Concatenating DDS PDUs and non-DDS PDUs by IpduM is out of the scope of this concept.
- **SoAdConnection status.** Each used `SoAdConnection` is assumed to be open.

  Note. A `SoAdConnection` can be explicitly open via `SoAd_OpenSoCon` or can be automatically open by `SoAd`. Refer to `SoAdSocketAutomaticSoConSetup` parameter description in [5] for details.
- **Application Tx PDU format in a bidirectional service interface elements.** The `DdsAppDataTxPdu` format is expected to be defined according to the rules in Chapter 7.1.2.2.
- **DdsDataWriter/DdsDataReader in service interface elements.** The `DdsDataWriter` and `DdsDataReader` used by a service interface element instance can handle at most one application PDU, respectively in transmission and in reception (See [CP_SWS_Dds_CONSTR_00960], [CP_SWS_Dds_CONSTR_00961]).

4.1.2 Limitations

- **DDS/CAN Data gateway:** Not supported
- **Shared-memory communication:** Not supported.
- **Immediate transmission:** Not supported.
- **Immediate reception:** Not supported.
- **Dds Transmit, Dds Trigger Transmit APIs support**: both functions `Dds_Transmit`, `Dds_TriggerTransmit` are always available, but only one of them must be used (e.g., `Dds_Transmit` is used instead of `Dds_TriggerTransmit` only if to be called by lower layer).
- **Software Cluster Connection Layer (SwCluC):** Not supported.
- **Multicore Distribution:** `Dds` module is assumed to be located and running in a single partition (no multiple sets of Tx/Rx main functions per partition).
- **Communication use cases:** Table 4.1 summarizes the supported Communication paths.
- **Application service control (e.g BswM support)**: In the case of SOME/IP, service control is necessary to allow the application to manage ECU state changes,

the availability of the communication channel, and event groups. In the case of Dds, the concept of an event group does not exist, but it could be extremely useful for managing other scenarios.

- **Dds_DeInit:** Not supported.
- **Communication stack status:** unavailability of underlying communication stack is not considered and Dds will continue retries (see [CP_SWS_Dds_01015])
- **Communication stack stop/restart:** stop/restart of underlying communication stack is currently up to implementation. To be finalized in the later releases, together with the limitation (Communication stack status) above.
- **End-to-end protection:** End-to-end protection is implemented by the Dds module (see Chapter 7.1.5) using the mechanisms defined by the DDS standard, (including counters synchronization). For this reason, the SW-C-RTE-DdXf-PduR-Dds section is not protected. In the next release, the design of this part will be reconsidered.
- **Domain Participant sharing between Service Instances:** A single DdsDomainParticipant cannot be shared between more than one DdsProvidedServiceInstance/DdsConsumedServiceInstance. In the current ECUC model this means that DdsDomainParticipant can aggregate only one DdsProvidedServiceInstance/DdsConsumedServiceInstance (see [CP_SWS_Dds_CONSTR_00970], [CP_SWS_Dds_CONSTR_00971]).

Short name:	Y/N
Signal-based with SenderReceiverInterface	Yes
Signal-based with ClientServerInterface	Yes
Signal-based with TriggerInterface	Yes
Service-oriented	Yes
Diagnostic	No
DLT	No
XCP	No
Mirror	No

Table 4.1: DDS supported Communication paths

4.1.3 Constraints

- **DDS-SOME/IP network interaction:** Dds and SOME/IP share the same bus type, so some mechanism to assure they don't interfere would be provided. This mechanism will mostly consist on UDP port choice: SOME/IP and DDS shall never share reception port numbers.
- **DDS-SOME/IP SoAd resource sharing:** Shall not be mix of DDS and SOME/IP (or any other potential protocol) communication on the same socket connection. A socket (or a set of sockets) is reserved for DDS only.

- **DDS/non-DDS PDU concatenation:** DDS and SOME/IP shall not share the same socket connection. A socket (or a set of sockets) shall be reserved to DDS only.
- **UDP Usage:** According to the OMG specification [2], the UDP/IP PSM shall be used for inter-ecu communications.

4.2 Applicability to car domains

This module is applicable all domains where DDS communication is required and/or beneficial.

5 Dependencies to other modules

This section describes the relations to other modules and files within the AUTOSAR basic software architecture. It contains brief descriptions of configuration information and services, which are required by the Dds module from other modules.

5.1 RTE (BSW Scheduler)

The RTE BSW Scheduler [6] calls the main functions of the Dds, which are necessary for the cyclic processes of the Dds.

5.2 PDU Router

The Dds module uses the PduR [7] as middle layer module.

5.3 StbM

OMG Standard states that each RTPS message sent by the originating Participant can include a timestamp. ([8] 8.3.2.2). It may be used by the receiving application to estimate the time offset between the clocks of the sending and receiving Participants (for instance in DESTINATION_ORDER QoS policy handling). The StbM_GetCurrentTime API shall be used to guarantee the needed precision ([8] 8.3.5.8, 9.4.2.9).

5.4 Default Error Tracer

In order to be able to report development or runtime errors, the Dds module has to have access to the error hook of the Default Error Tracer [9].

5.5 Crypto Service Manager

In order to support Security capabilities (e.g., Key management, Message Authentication Code generation and verification), the Dds shall use the Crypto Service Module API [10]: The Dds requires:

- the MAC-generate interface (Csm_MacGenerate) to generate MAC to be added to messages to be sent;

- the MAC-verify interface (Csm_MacVerify) to check MAC of received messages.

5.6 Cyclic Redundancy Check

In order to support Safety capabilities, Dds uses the CRC Library [11].

The Dds requires the Crc_CalculateCRC32 or the Crc_CalculateCRC64 APIs to calculate CRC to be added to messages to be sent or to be checked for received messages.

5.7 Socket Adaptor

The Socket Adaptor [5] hands over service requests between the Ethernet Stack and the Dds. Dds module interacts with SoAd updating remote endpoint information (IP address and port number) in predefined SoAdSocketConnection.

6 Requirements Tracing

The following tables reference the requirements specified in [1, RS-DDS] 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.

Requirements that are not fulfilled by this document are linked to [SWS_Dds_NA_00000].

Requirement	Description	Satisfied by
[FO_RS_Dds_00001]	DDS Compliance	[CP_SWS_Dds_00734] [CP_SWS_Dds_00736] [CP_SWS_Dds_00859] [CP_SWS_Dds_00900] [CP_SWS_Dds_00901] [CP_SWS_Dds_00902] [CP_SWS_Dds_00903] [CP_SWS_Dds_00904] [CP_SWS_Dds_00905] [CP_SWS_Dds_00906] [CP_SWS_Dds_00907] [CP_SWS_Dds_00908] [CP_SWS_Dds_00909] [CP_SWS_Dds_00921] [CP_SWS_Dds_00925] [CP_SWS_Dds_00926] [CP_SWS_Dds_00927] [CP_SWS_Dds_00928] [CP_SWS_Dds_00930] [CP_SWS_Dds_00931] [CP_SWS_Dds_01010] [CP_SWS_Dds_01011] [CP_SWS_Dds_01012] [CP_SWS_Dds_01013] [CP_SWS_Dds_01014] [CP_SWS_Dds_01015] [CP_SWS_Dds_01019] [CP_SWS_Dds_01020] [CP_SWS_Dds_01031]
[FO_RS_Dds_00002]	DDS standard serialization rules	[CP_SWS_Dds_00726]
[FO_RS_Dds_00004]	DDS payload serialization rules	[CP_SWS_Dds_00728] [CP_SWS_Dds_00729] [CP_SWS_Dds_00730] [CP_SWS_Dds_00731] [CP_SWS_Dds_00734] [CP_SWS_Dds_00735] [CP_SWS_Dds_00736]
[FO_RS_Dds_00005]	DDS Quality of Service	[CP_SWS_Dds_00763] [CP_SWS_Dds_00764] [CP_SWS_Dds_00773] [CP_SWS_Dds_00832] [CP_SWS_Dds_00833] [CP_SWS_Dds_01001]
[FO_RS_Dds_00006]	The DDS AUTOSAR components receive unserialized data	[CP_SWS_Dds_00728] [CP_SWS_Dds_00729] [CP_SWS_Dds_00730] [CP_SWS_Dds_00731] [CP_SWS_Dds_00734] [CP_SWS_Dds_00735] [CP_SWS_Dds_00736]
[FO_RS_Dds_00007]	Type Definition	[CP_SWS_Dds_00728] [CP_SWS_Dds_00729] [CP_SWS_Dds_00730] [CP_SWS_Dds_00731] [CP_SWS_Dds_00735] [CP_SWS_Dds_00801] [CP_SWS_Dds_00802]
[FO_RS_Dds_00009]	Security mechanism	[CP_SWS_Dds_00750] [CP_SWS_Dds_00752] [CP_SWS_Dds_00753] [CP_SWS_Dds_00756] [CP_SWS_Dds_00758] [CP_SWS_Dds_00773] [CP_SWS_Dds_00832] [CP_SWS_Dds_00833]
[FO_RS_Dds_00010]	Safety mechanism	[CP_SWS_Dds_00761] [CP_SWS_Dds_00762] [CP_SWS_Dds_00763] [CP_SWS_Dds_00764] [CP_SWS_Dds_00766] [CP_SWS_Dds_00769] [CP_SWS_Dds_00773] [CP_SWS_Dds_00832] [CP_SWS_Dds_00833]

▽

△

Requirement	Description	Satisfied by
[FO_RS_Dds_00015]	Publish	[CP_SWS_Dds_00772] [CP_SWS_Dds_00773] [CP_SWS_Dds_00828] [CP_SWS_Dds_00829] [CP_SWS_Dds_00830] [CP_SWS_Dds_00832] [CP_SWS_Dds_00835] [CP_SWS_Dds_00837] [CP_SWS_Dds_00838] [CP_SWS_Dds_00843] [CP_SWS_Dds_00851] [CP_SWS_Dds_00852] [CP_SWS_Dds_00854] [CP_SWS_Dds_00855] [CP_SWS_Dds_00859] [CP_SWS_Dds_00871] [CP_SWS_Dds_00872] [CP_SWS_Dds_00873] [CP_SWS_Dds_00881] [CP_SWS_Dds_00882] [CP_SWS_Dds_00883] [SWS_Dds_00844] [SWS_Dds_10001] [SWS_Dds_10002] [SWS_Dds_10003] [SWS_Dds_10004] [SWS_Dds_10005] [SWS_Dds_10006] [SWS_Dds_10007] [SWS_Dds_10008] [SWS_Dds_10009] [SWS_Dds_10010] [SWS_Dds_10011]
[FO_RS_Dds_00016]	Subscribe	[CP_SWS_Dds_00772] [CP_SWS_Dds_00773] [CP_SWS_Dds_00825] [CP_SWS_Dds_00826] [CP_SWS_Dds_00827] [CP_SWS_Dds_00832] [CP_SWS_Dds_00834] [CP_SWS_Dds_00836] [CP_SWS_Dds_00841] [CP_SWS_Dds_00861] [CP_SWS_Dds_00862] [CP_SWS_Dds_00863] [CP_SWS_Dds_00864] [CP_SWS_Dds_00873]
[FO_RS_Dds_00017]	Transport protocol	[CP_SWS_Dds_00726]
[FO_RS_Dds_00019]	RTPS message encapsulation	[CP_SWS_Dds_00726] [CP_SWS_Dds_00734] [CP_SWS_Dds_00736]
[FO_RS_Dds_00020]	RTPS message decapsulation	[CP_SWS_Dds_00726] [CP_SWS_Dds_00734] [CP_SWS_Dds_00736]
[FO_RS_Dds_00030]	DDS Event communication support	[CP_SWS_Dds_00900] [CP_SWS_Dds_00902]
[FO_RS_Dds_00031]	DDS Method communication support	[CP_SWS_Dds_00904] [CP_SWS_Dds_00905] [CP_SWS_Dds_00906] [CP_SWS_Dds_00907] [CP_SWS_Dds_00921] [CP_SWS_Dds_00925] [CP_SWS_Dds_00926] [CP_SWS_Dds_00927] [CP_SWS_Dds_00928] [CP_SWS_Dds_00930] [CP_SWS_Dds_00931]
[FO_RS_Dds_00032]	DDS error handling of bi-directional method communication support	[CP_SWS_Dds_00905] [CP_SWS_Dds_00908] [CP_SWS_Dds_00909]
[FO_RS_Dds_00033]	DDS shall support field communication with getters, setters and notification events.	[CP_SWS_Dds_00901] [CP_SWS_Dds_00903] [CP_SWS_Dds_00908] [CP_SWS_Dds_00909] [CP_SWS_Dds_00925] [CP_SWS_Dds_00926] [CP_SWS_Dds_00927] [CP_SWS_Dds_00928] [CP_SWS_Dds_00930] [CP_SWS_Dds_00931]
[FO_RS_Dds_00040]	DDS shall optionally support SPDP according to DDS-RTPS specification	[CP_SWS_Dds_00737] [CP_SWS_Dds_00738] [CP_SWS_Dds_01010] [CP_SWS_Dds_01013] [CP_SWS_Dds_01014] [CP_SWS_Dds_01015] [CP_SWS_Dds_01019] [CP_SWS_Dds_01020] [CP_SWS_Dds_01031]
[FO_RS_Dds_00041]	DDS shall optionally support SEDP according to DDS-RTPS specification	[CP_SWS_Dds_00737] [CP_SWS_Dds_00738] [CP_SWS_Dds_01011] [CP_SWS_Dds_01013] [CP_SWS_Dds_01014] [CP_SWS_Dds_01015] [CP_SWS_Dds_01019] [CP_SWS_Dds_01020] [CP_SWS_Dds_01031]

▽

△

Requirement	Description	Satisfied by
[FO_RS_Dds_00042]	DDS shall support the Domain Participant USER_DATA QoS policy, if Service Discovery as described by AUTOSAR DDS Service Discovery Protocol specification is to be supported	[CP_SWS_Dds_00737] [CP_SWS_Dds_00738] [CP_SWS_Dds_01012] [CP_SWS_Dds_01013] [CP_SWS_Dds_01014] [CP_SWS_Dds_01015] [CP_SWS_Dds_01019] [CP_SWS_Dds_01020] [CP_SWS_Dds_01031]
[FO_RS_Dds_00043]	DDS shall support the Publisher/ Subscriper PARTITION QoS policy, if Partition-based Resource Identification as described by AUTOSAR DDS Service Discovery Protocol specification is to be supported	[CP_SWS_Dds_00737] [CP_SWS_Dds_00738] [CP_SWS_Dds_01012] [CP_SWS_Dds_01013] [CP_SWS_Dds_01014] [CP_SWS_Dds_01015] [CP_SWS_Dds_01019] [CP_SWS_Dds_01020] [CP_SWS_Dds_01031]
[SRS_BSW_00003]	All software modules shall provide version and identification information	[CP_SWS_Dds_00831]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[CP_SWS_Dds_00810] [CP_SWS_Dds_00811] [CP_SWS_Dds_00812] [CP_SWS_Dds_00813]
[SRS_BSW_00318]	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	[CP_SWS_Dds_00831]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[CP_SWS_Dds_00852] [CP_SWS_Dds_00881]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[CP_SWS_Dds_00823] [CP_SWS_Dds_00824]
[SRS_BSW_00374]	All Basic Software Modules shall provide a readable module vendor identification	[CP_SWS_Dds_00831]
[SRS_BSW_00379]	All software modules shall provide a module identifier in the header file and in the module XML description file.	[CP_SWS_Dds_00831]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[CP_SWS_Dds_00832] [CP_SWS_Dds_00833]
[SRS_BSW_00385]	List possible error notifications	[CP_SWS_Dds_00832] [CP_SWS_Dds_00833]
[SRS_BSW_00402]	Each module shall provide version information	[CP_SWS_Dds_00831]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[CP_SWS_Dds_00802] [CP_SWS_Dds_00810]
[SRS_BSW_00411]	AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of API(s)	[CP_SWS_Dds_00831]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[CP_SWS_Dds_00810]
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[CP_SWS_Dds_00823] [CP_SWS_Dds_00824]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[CP_SWS_Dds_00823] [CP_SWS_Dds_00824]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[CP_SWS_Dds_00823] [CP_SWS_Dds_00824]
[SRS_BSW_00478]	Timing limits of main functions	[CP_SWS_Dds_00823] [CP_SWS_Dds_00824]
[SRS_BSW_00496]	BSW modules shall be able to handle PDU meta data	[CP_SWS_Dds_00720] [CP_SWS_Dds_00721] [CP_SWS_Dds_00722] [CP_SWS_Dds_00723]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Overview

The Dds module implements all the interface logic (Entity management, QoS, etc.) and the DDS-RTPS standard layer [8] for DDS communication.

The Dds module is a container that encapsulates the implementation of a full-fledged middleware in accordance with the OMG standards and defines an interface based on PDUs typical of the AUTOSAR Classic Platform. Due to its middleware nature, the application PDUs must essentially carry the data produced by the application unaltered, so that they can later be processed by the logic residing entirely within the module (See Figure 7.1).

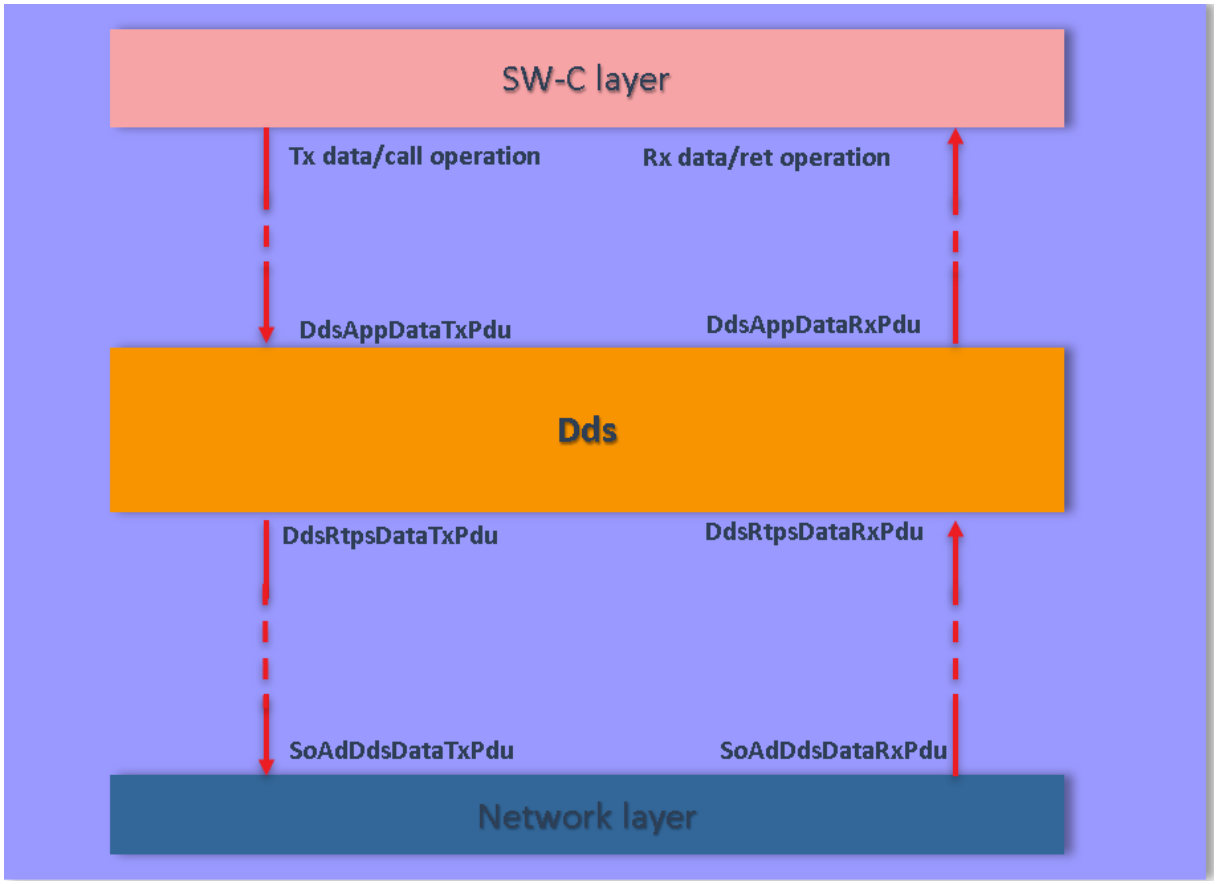


Figure 7.1: The Dds functional scheme in AUTOSAR Classic Platform

It is composed by several functional aspects:

- RTPS protocol Serialization
- RTPS protocol Deserialization
- Data filtering
- Data reordering

- Data persistency
- Data re-transmission
- Security
- E2E protection
- Remote Procedure Call
- Participants, Endpoints and Service Instance discovery procedures.

7.1.1 Communication paths

Dds supports following AUTOSAR Classic Platform Communication Paths:

- **Signal-based with** SenderReceiverInterface
- **Signal-based with** ClientServerInterface
- **Signal-based with** TriggerInterface

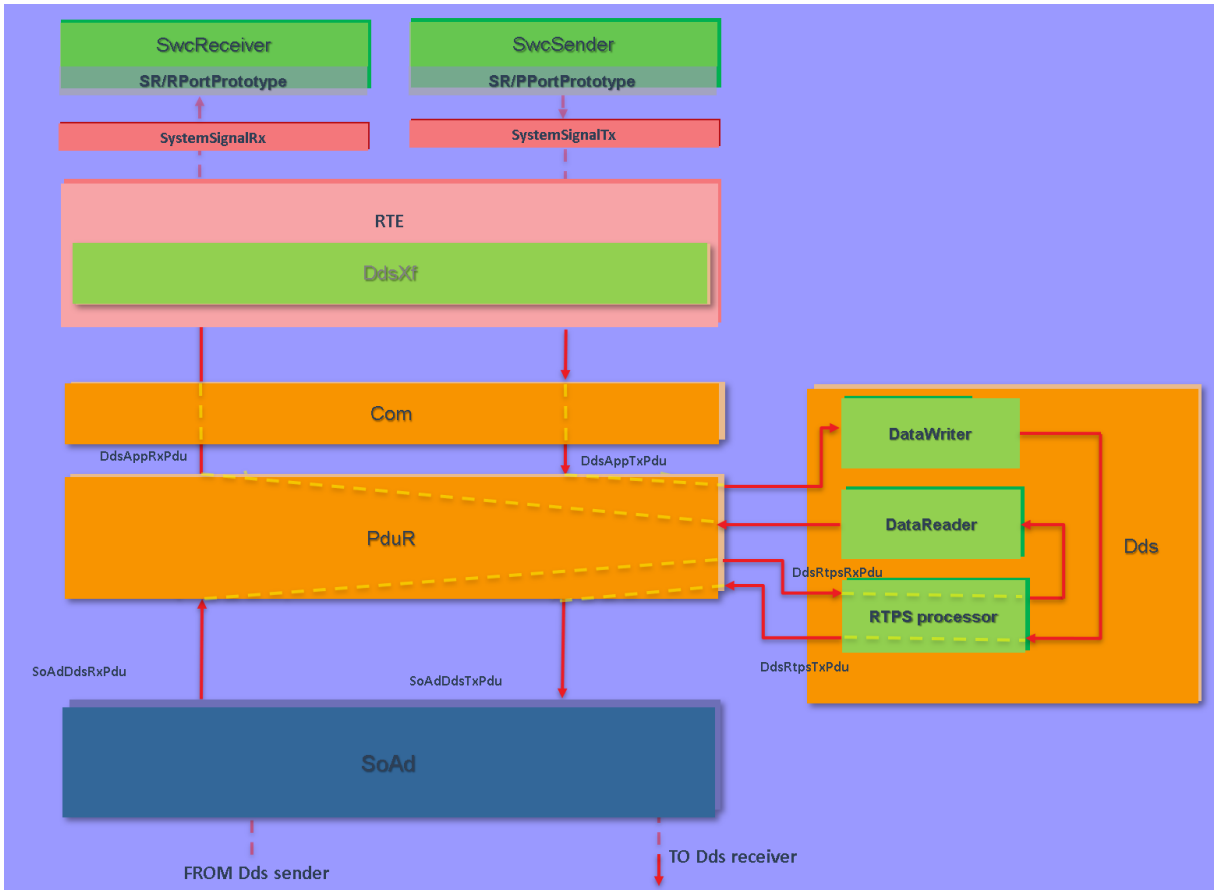


Figure 7.2: SenderReceiverInterface communication path

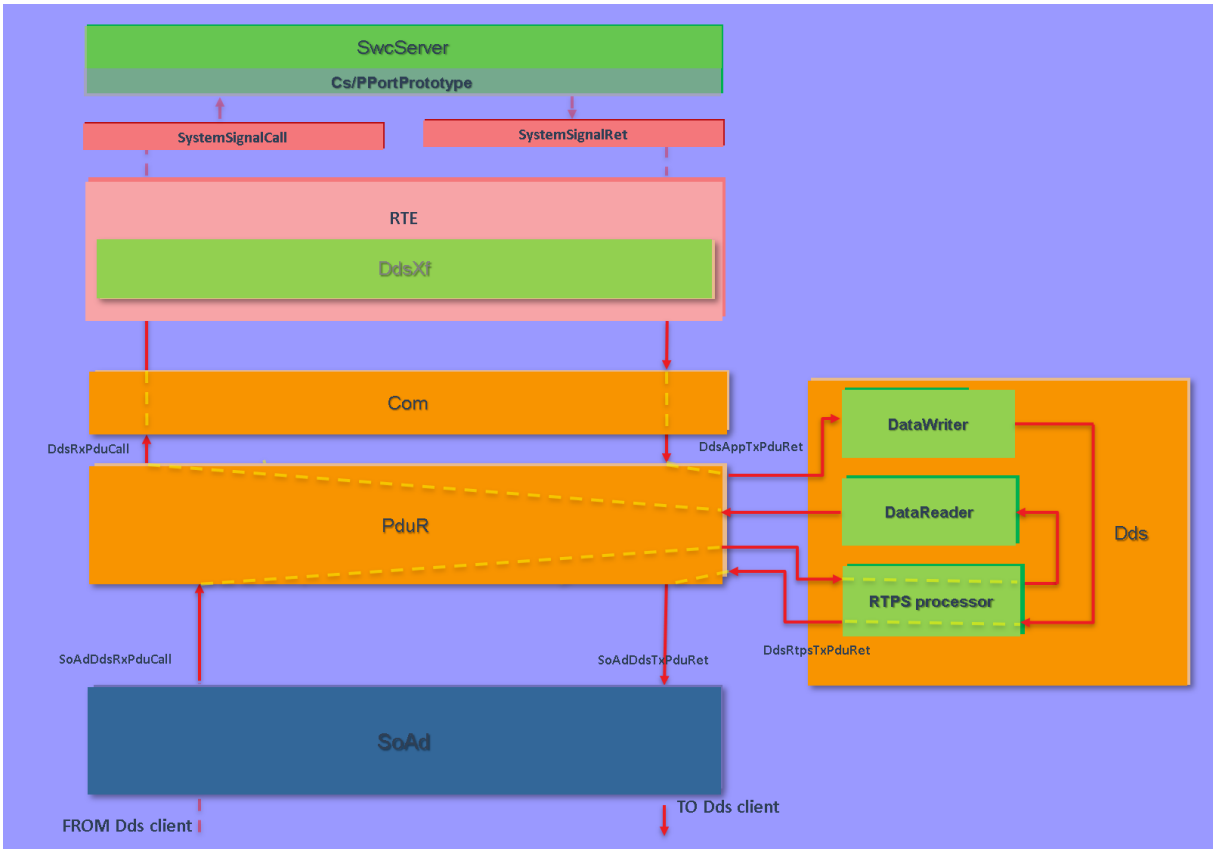


Figure 7.3: ClientServerInterface. Server side

Basically `Dds` consumes the application PDU and, according to the internal logic, can decide to produce and transmit a new RTPS PDU through the transport layer.

The `DataWriter` and `DataReader` represent internal functional blocks showing the mapping between a PDU and the related DDS middleware object (they are modeled in the ECUC by some `DdsDataWriter`/`DdsDataReader`). The RTPS processor represents the logic of module.

Considering the transmission direction, the application data is fed into the Dds Transformer, which simply copies the `VariableDataPrototype` and `ClientServerOperation` into a buffer that will serve as the content of the `DdsRtpsDataTxPdu`. In the reverse direction, the `VariableDataPrototype` and `ClientServerOperation` results are similarly extracted from the received PDU.

The `TriggerInterface` use case can be considered as a special case of the `SenderReceiverInterface` where the data is empty, so in this case the Dds Transformer is not used.

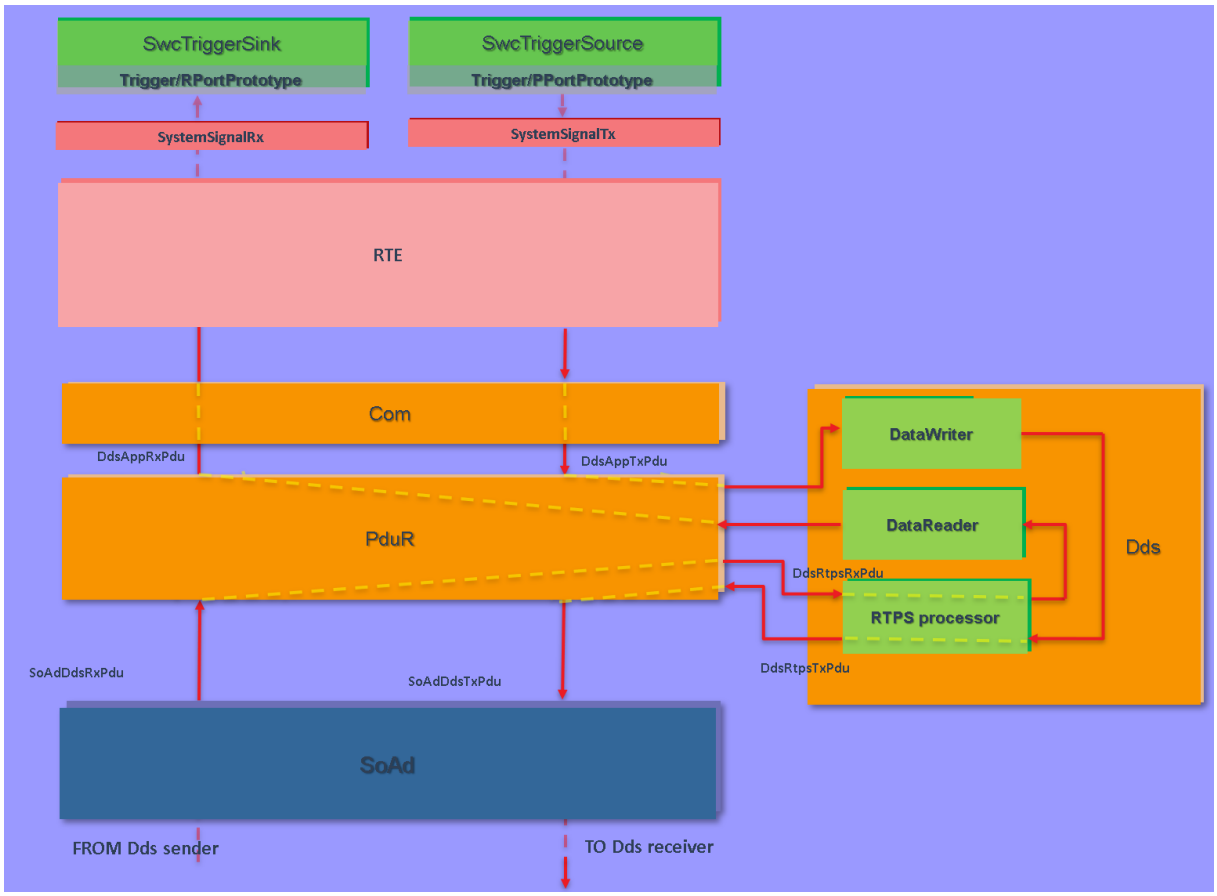


Figure 7.4: TriggerInterface communication path

In general case the structured data is passed to DDS middleware that performs its own processing, serializes resulting data into an RTPS packet and then saves the final RTPS packet into the lower PDU (DdsRtpsDataTxPdu or DdsRtpsMulticast-DataTxPdu). This PDU is forwarded to the PduR and then to the transport layer (e.g SoAd).

Thereby the DDS middleware internal policies decide which target PDU is used.

Dds interacts with the PduR only offering a PDU-based interface for the incoming and outgoing PDUs. The transport layer provides a set of connections suitable to enable the DDS communication.

Connections can be even automatically configured by the **Dds Dynamic discovery engine**.

7.1.2 Service Oriented Architecture

In AUTOSAR Classic Platform SOME/IP [12] is the protocol historically designed to handle SOA features. This specification introduces the DDS as a way to implement Service-Oriented communication; a brief introduction on SOA is described in [1].

DDS can realize the middleware for SOA applications implementing the different communication styles, **Events**, **Methods** and **Fields**, using three foundational AUTOSAR PortInterfaces:

- SenderReceiverInterface
- ClientServerInterface
- TriggerInterface

AUTOSAR Classic Platform introduces the service interface in [13, chapter 10.1 AUTOSAR_CP_TPS_SystemTemplate].

Each service interface element instance is modeled with a VariableDataPrototype, a ClientServerOperation or a Trigger and it is uniquely identified with one or two ISignalIPdu.

Service interface element	Represented by	Identified by
Event	VariableDataPrototype in a SenderReceiverInterface	one ISignalIPdu
Request-Response Method	ClientServerOperation in a ClientServerInterface	one ISignalIPdu for request and one ISignalIPdu for reply
Fire-Forget Method with arguments	VariableDataPrototype in a SenderReceiverInterface	one ISignalIPdu
Fire-Forget Method with no arguments	Trigger in a TriggerInterface	one ISignalIPdu
Field Notifier	VariableDataPrototype in a SenderReceiverInterface	one ISignalIPdu
Field Getter	ClientServerOperation in a ClientServerInterface with ArgumentDataPrototype with direction out	one ISignalIPdu for request and one ISignalIPdu for reply
Field Setter	ClientServerOperation in a ClientServerInterface with ArgumentDataPrototype with direction in	one ISignalIPdu for request and one ISignalIPdu for reply

Table 7.1: Representation of the service interface model in AUTOSAR Classic Platform

Each service interface element could be gathered in a Collection class following specific rules as stated in reference. This kind of Collection is able to mimic the Adaptive Platform meaning of ServiceInterface.

7.1.2.1 Service interface elements with Dds

In this specification, System template elements (DdsCpProvidedServiceInstance and DdsCpConsumedServiceInstance) and ECU configuration template elements (DdsProvidedServiceInstance and DdsConsumedServiceInstance) are de-

rived from the Service Interface Collection concept, in order to model and configure Service Instances realising such Service Interfaces.

Table 7.2 shows the mappings of each service interface element with an object defined in the Dds ECU configuration model.

Two different sets of service interface elements are defined:

- **Bidirectional:** a Request-Response communication style is implemented between two ECUs
- **Unidirectional:** a one-way data sending communication style is used by an ECU toward another ECU.

Service interface element	Represented by	Topics data types	Communication style
Event	DdsServiceEventSender DdsServiceEventReceiver	[FO_PRS_DDS_00101]	Unidirectional
Request-Response Method	DdsServiceMethodsRequestReplyMethod	[FO_PRS_DDS_00301] [FO_PRS_DDS_00302]	Bididirectional
Fire-Forget Method with arguments	DdsServiceMethodsFireAndForgetMethod	[FO_PRS_DDS_00301]	Unidirectional
Fire-Forget Method with no arguments	DdsServiceMethodsFireAndForgetMethod	[FO_PRS_DDS_00301]	Unidirectional
Field Notifier	DdsServiceFieldsFieldNotifierSender DdsServiceFieldsFieldNotifierReceiver	[FO_PRS_DDS_00401]	Unidirectional
Field Getter	DdsServiceFieldsFieldServer DdsServiceFieldsFieldClient	[FO_PRS_DDS_00408] [FO_PRS_DDS_00409]	Bidirectional
Field Setter	DdsServiceFieldsFieldServer DdsServiceFieldsFieldClient	[FO_PRS_DDS_00408] [FO_PRS_DDS_00409]	Bidirectional

Table 7.2: Mapping of service interface element in Dds ECU configuration model

Each Topic used to realize SOA features should be mapped to a well-defined Topic data type based on the service interface element it models. Bidirectional service interface elements rely on two Topics, while unidirectional elements rely on only one.

For each element, by comparing the two previous tables, it is evident that there is a one-to-one relationship between each individual PDU and each individual Topic. Moreover, given the following assertions: first, that a generic element manages either a DdsDataWriter, or a DdsDataReader, or a combination of both, and second, that a generic DdsDataWriter/DdsDataReader, can be associated with only one Topic, it follows that the generic DdsDataWriter/DdsDataReader, can refer to only a single PDU. This condition is formalized in [CP_SWS_Dds_CONSTR_00960], [CP_SWS_Dds_CONSTR_00961].

In Appendix D a more detailed description of the SOA architecture implementation with DDS is described. The DDS middleware building blocks and the supported PortInterfaces are highlighted.

7.1.2.2 Bidirectional service interface elements

The **Bidirectional service interface elements** are the Request-Response methods and Getter/Setter operation Fields defined respectively in Chapter 7.2.1.8.4 and in Chapter 7.2.1.8.5.2.

They rely on the Remote Procedure Call protocol specified in [14, DDS RPC Specification] and in the [15, AUTOSAR DDS Protocol Specification] according to [CP_SWS_Dds_00904], [CP_SWS_Dds_00905], [CP_SWS_Dds_00908], [CP_SWS_Dds_00909].

In order to fullfill all the requirements in Chapter 7.2.1.8.1, the `Dds` requires that the format of a certain `DdsAppDataTxPdu` referenced by one of:

- `DdsServiceMethodsRequestReplyMethod.DdsDataWriterRef`
- `DdsServiceFieldsFieldServerGetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldServerSetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldClientGetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldClientSetter.DdsDataWriterRef`

is defined according to [CP_SWS_DdsXf_00143], [CP_SWS_DdsXf_00169], as showed in Figure 7.5.

This type of service element is based on the use of a `ClientServerOperation` and therefore on the management of the `requestId` field required to distinguish between different requests of the same service type, in accordance with the semantics defined by the [6, chapter 4.3.2 in RTE specification].

The rules to do that are specified by [CP_SWS_Dds_00925], [CP_SWS_Dds_00926], [CP_SWS_Dds_00927], [CP_SWS_Dds_00928].

More functional details can be found in Appendix E.

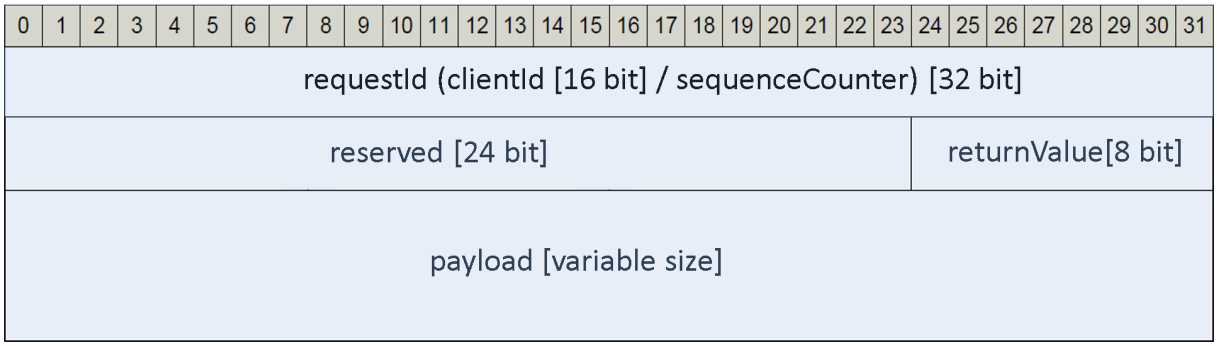


Figure 7.5: The expected application PDU format in case of request and response service elements handling

7.1.3 QoS Management

The Dds could support a subset (even empty) of QoS policies. There is no mandatory QoS to be implemented. It is vendor-specific which QoS policy is actually supported.

Each entity could define its own set of supported QoS policy, by static configuration.

7.1.3.1 TRANSPORT_PRIORITY QoS mapping

The DDS standard defines the TRANSPORT_PRIORITY QoS for a certain DataWriter. Its purpose is to allow the application to take advantage of transports capable of sending messages with different priorities (any value within the range of a 32-bit signed integer may be chosen; higher values indicate higher priority). In AUTOSAR CP the information closest to the concept of transport priority is the value of SoAdSocketFramePriority defined in SoAdSocketConnectionGroup parent object of the SoAd module. This specifies the priority of the Ethernet frame handled for all the SoAdSocketConnections owned by the related SoAdConnectionGroup.

Unfortunately there is no direct link between the application level of the Dds module (e.g., the upper layer PDU and the DdsDataWriter) and its transport level (e.g., the lower layer PDU).

In other words: the Dds module selects at runtime the lower layer PDU based on the configuration of the according DdsDataWriter. If any, the Dds module should select a DdsRtpsDataTxPdu that belongs to a SoAdSocketConnectionGroup configured with the needed SoAdSocketFramePriority.

Note. The TRANSPORT_PRIORITY QoS is considered just an hint for the underlying transport protocol. The policy depends on the ability of the underlying transports to set a priority on the messages they send.

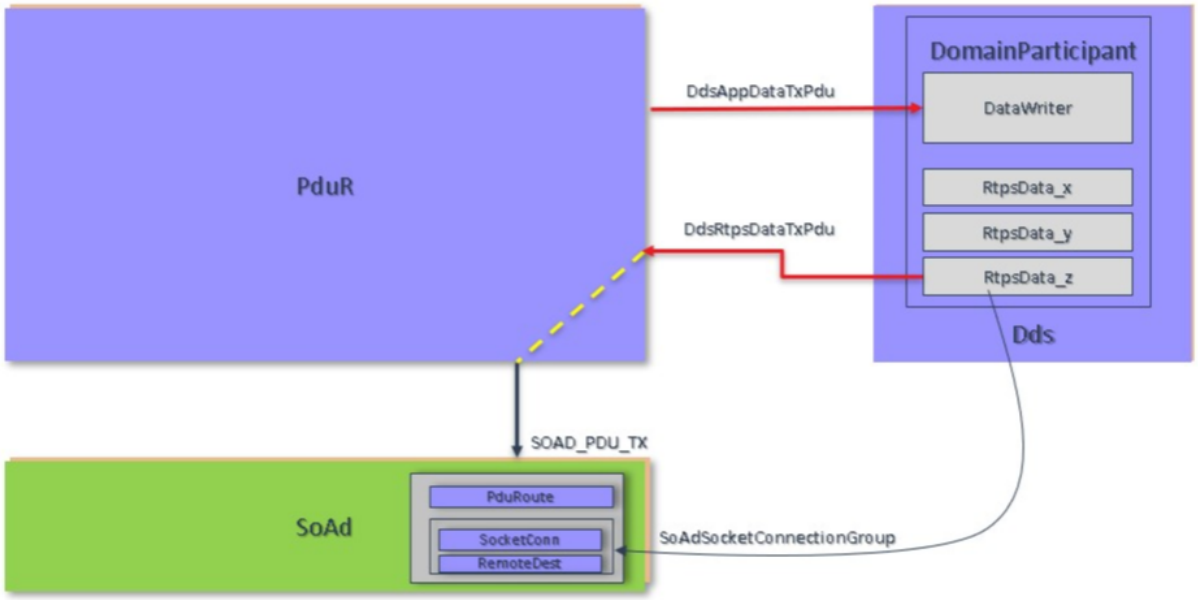


Figure 7.6: DataWriter Transport priority QoS value

7.1.4 Security Mechanisms

Opening up a communication path between AP and CP, and even between CP and non-AUTOSAR platforms, could involve security risks, so it may require the usage of some security mechanisms.

The `Dds` Module guarantees some security mechanism by the usage of DDS Security Specification [16]. The usage of this specification is necessary to guarantee the interoperability with other DDS systems, both with AP (where DDS- Security is already in use) and in non-AUTOSAR systems. Implementing this specification, however, could be really resource consuming. In particular, to be used on a slow microcontroller, these features would need hardware acceleration. To overcome this issue, a subset of DDS-security functionalities which guarantee a minimum security level has been selected.

At this stage, implementing DDS-Security aims to guarantee message authentication, data integrity and group authentication. Security mechanism can be enabled or disabled at configuration time. If enabled, all security parameters must be statically configured at pre-compile time. For details on security parameters configuration, please refer to Chapter 10.2.3.3.1.4.

If configured, a Message Authentication Code (MAC) of the entire RTPS message is added. The AUTOSAR `Csm` module is used for key management and MAC calculation. Which algorithm to be used is configurable (choosing from supported ones).

The keys used for hash algorithms are symmetric keys shared between entities associated to a `DomainParticipant`, so authentication is done at `DomainParticipant` level (not of single `Publisher`/`Subscriber`, not of single `DataWriter`/`DataReader`). The symmetric key to be used for a specific DomainParticipant shall be managed directly by `Csm`, which should provide a handle to DDS to use its services.

For the above mentioned purposes, the DDS **Cryptographic Plugin** is used, which offers an interface to protect the whole RTPS message. The resulting RTPS message, after security is applied, is shown in the picture Figure 7.7 below.

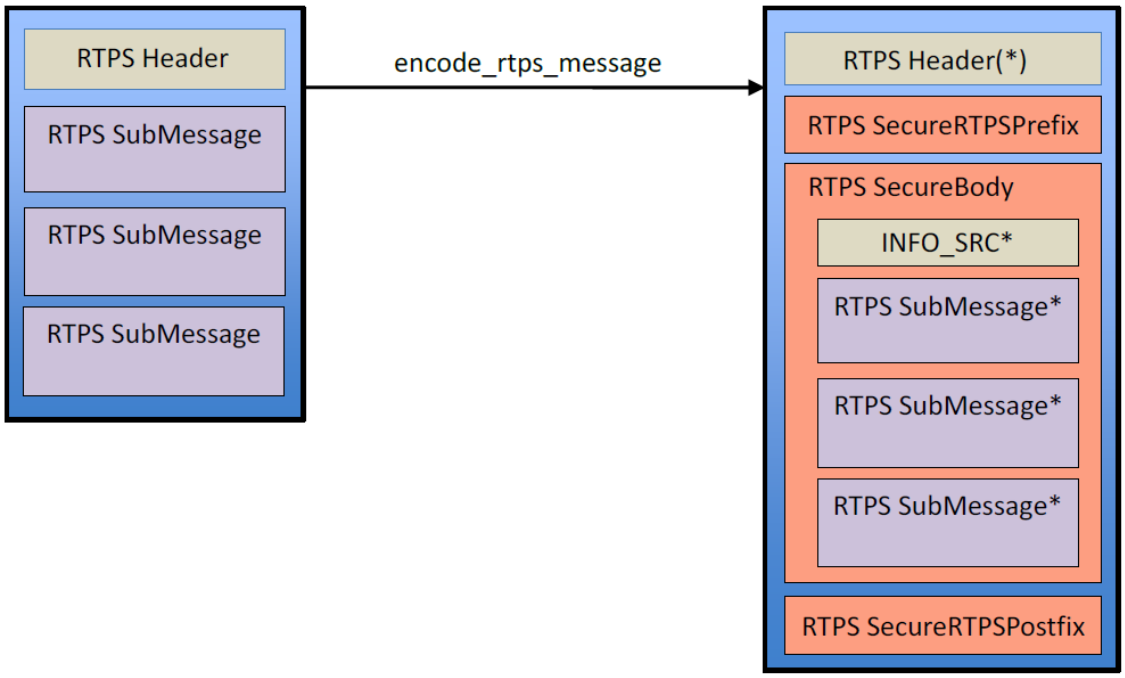


Figure 7.7: Dds RTPS message encoding

7.1.5 Safety Mechanisms

According to ISO 26262, there is a set of faults that can be considered on communication link between senders and receivers executed in different software partitions or ECUs.

The concept of end-to-end protection assumes that safety-related data exchange shall be protected at runtime against the effects of faults within the communication link.

The DDS Specification has intrinsic safety mechanisms (Counters, CRCs, QoS policies) that can be used to support a safety argument.

For a complete list of possible faults as defined in [17] to be addressed in the pursuit of functional safety, and the mechanisms DDS offers to support them, please refer to section 6.12 of [15] and 7.2.3.

7.1.6 Discovery mechanism

In this document, the term **Dds Dynamic Discovery engine** indicates the `Dds`'s ability to obtain, at run-time, information about the presence of remote Participants, remote Endpoints, and their relationships in terms of service instances.

This can be done through the support of three different discovery mechanisms using unicast and multicast paradigms:

- **Simple Participant Discovery Protocol** (SPDP) [8]
- **Simple Endpoint Discovery Protocol** (SEDP) [8]
- **Service Discovery protocol** [18]

The DDS protocol supports both unicast and multicast protocols for discovery traffic as well as user traffic [8].

Note. IP multicast in DDS user traffic Using IP multicast in DDS allows a single message to be sent simultaneously to multiple subscribers. This approach not only minimizes bandwidth usage and simplifies traffic management but also enhances scalability by eliminating the need for individual message transmissions, thereby reducing latency through concurrent data delivery. Additionally, when combined with DDS QoS settings, multicast optimizes network resources to maintain robust, high-performance communications even in complex or real-time automotive systems. A typical scenario. A DDS DataWriter can transmit a single message on a specific Topic using a multicast group, thereby ensuring the simultaneous distribution of the message to all DDS DataReaders subscribed to it and optimizing both bandwidth usage and synchronization.

The `Dds` module supports the SPDP protocol via multicast and the SEDP protocol via unicast. This is achieved through the definition of distinct sets of PDUs.

Three different network scenarios are allowed:

- **All dynamic:** Participant Discovery, Endpoint Discovery and Service Instance Discovery (SPDP enabled, SEDP enabled and Service Instance Discovery).
- **Partially Dynamic:** Static participant discovery but everything else is dynamic (Static remote Participants, Dynamic Endpoints and Service Instances. SPDP enabled for matching, SEDP enbaled and Service Instance Discovery).
- **All static:** configured Participants, Endpoints and Service Instances (SPDP disabled, SEDP disabled and statically configured Service Instances.)

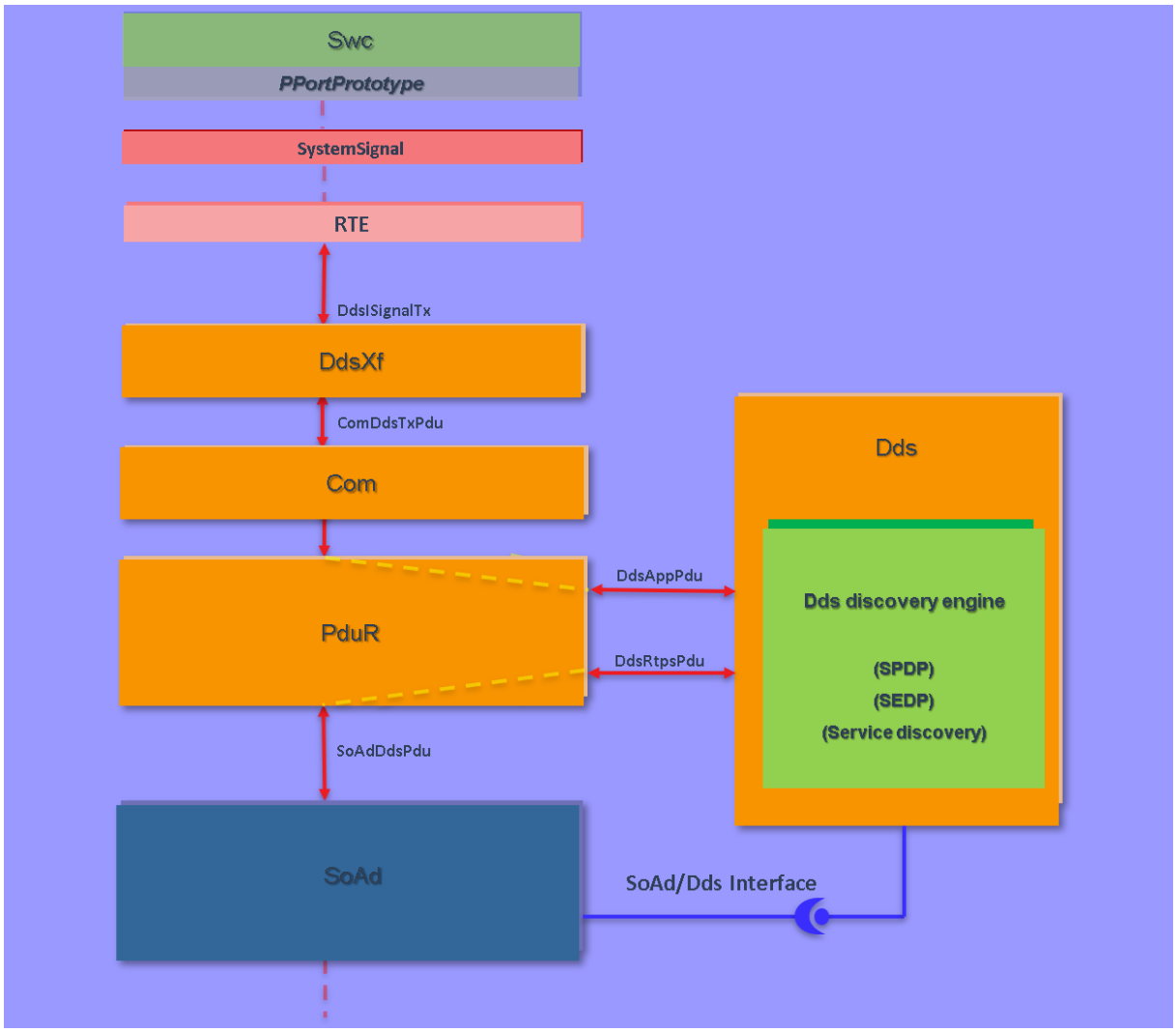


Figure 7.8: The Dds Discovery engine

Within an AUTOSAR Classic configuration, the transport layer shall ensure the configuration of a predefined set of connections, supporting unicast communication and, where applicable, multicast communication. More details could be found in Chapter F.1.

In a fully static configuration, the `Dds` module does not manage these connections directly; instead, it simply utilizes them via RTPS PDUs that are explicitly allocated to each `DdsRemoteDomainParticipant` in the unicast case, and to all `DdsRemoteDomainParticipant`s sharing the same `DdsDomainId` in the multicast case.

In network environments with fully or partially dynamic configurations, the `Dds` module must exert direct control over the connections. Two operational scenarios emerge:

- **All Dynamic:** The `Dds` module configures an available `SoAdConnection` by associating it with a `<IP address, port>` pair corresponding to the discovered remote Domain Participant, both for data traffic and SEDP meta-traffic. This establishes a unique mapping between the `DdsRtpsDataTxPdu` and `DdsRtpsDpDataTxPdu` entities and their respective `SoAdConnection`s.

- **Partially Dynamic:** In this scenario, the identifiers of the remote Domain Participants with which the generic local Domain Participant is expected to communicate are preconfigured, whereas the endpoints (remote Data Writers, remote Data readers, and their associated QoS parameters) are not known in advance. The SPDP and SEDP discovery procedures must still be enabled, and the resulting connection management follows the same approach as in the fully dynamic scenario. The key difference lies in the fact that only discovered remote Domain Participants whose identifiers match the preconfigured ones shall be accepted.

Note. It is useful here to recall that the DDS protocol standard [8] defines the single `DdsDomainParticipant` as the only entity to which an `<IP address, port>` pair can be associated; the various endpoints associated with it are referable only within the semantics of the protocol.

From the **behaviour** point of view, the `Dds` module implements the three discovery mechanisms as a background process appearing on the scene just when a user connection or an SEDP connection needs to be updated.

Main properties are:

- **Start-up**. The discovery engine starts at the beginning after the invocation of `Dds_Init`
- **External control**. No external (e.g SW-C by means of `BswM`) control is supported.
- **Connections handling**. Run-time control of the `SoAd` local connections is supported.
- **Discovery traffic (i.e meta-traffic)**. The meta-traffic incurred by every protocol employed is carried by the RTPS PDUs.

In Chapter F.1.1 an example of connections configuration is showed.

7.1.6.1 Socket connections control

The interaction of `Dds` with the single connection occurs through the interface exported by `SoAd` to set the `<IP address, port>` pair. This makes

sense for the single participant that must use SPDP or SEDP in accordance with the requirements [CP_SWS_Dds_01010] and [CP_SWS_Dds_01011].

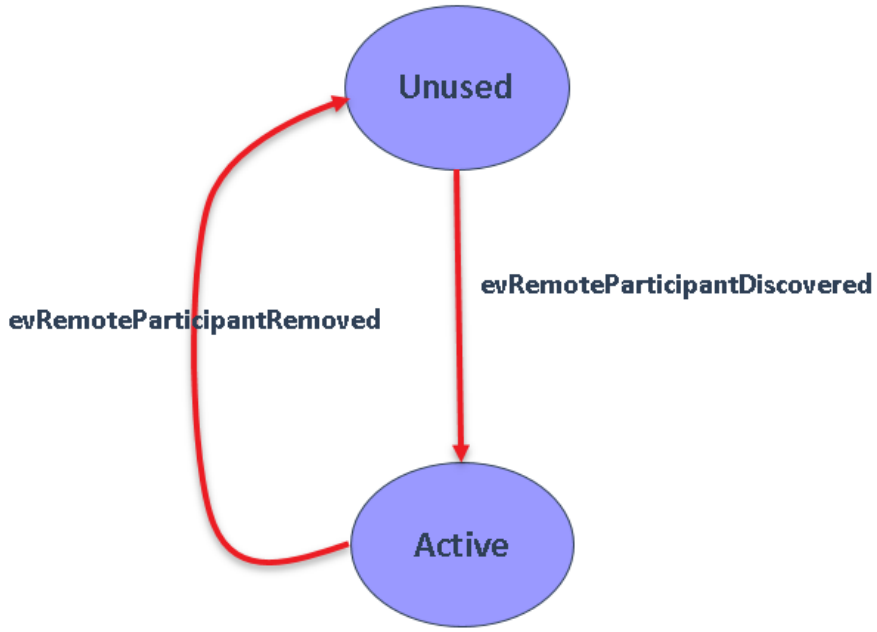


Figure 7.9: Finite state machine describing the control of a SoAdConnection

The dynamic behavior is schematically represented by the finite state machine in Figure 7.9. It describes the evolution of the connectivity status of a single `SoAdConnection` controlled by the Discovery engine.

- **FsmState**:`Unused`. The `SoAdConnection` cannot be used to send message, no remote address has been set.
- **FsmState**:`Active`. The `SoAdConnection` is configured with a valid remote IP address and port. It can be used to send message.
- **FsmEvent**:`evRemoteParticipantDiscovered`. A new remote Domain Participant has been discovered by the SPDP (a.k.a a new IP address and port are ready to be set in a connection) Each `DdsDomainParticipant` that wants to communicate with it must perform the following actions:
  - assign the pair `<remote IP address, remote port>` to an unused `SoAdConnection` identified by a specific `SoConnId`. Once assigned, the `SoAdConnection` becomes active.
  - maintain the association between the `SoConnId` and the pair `<remote IP address, remote port>`.

- **FsmEvent**:`evRemoteParticipantRemoved`. A known remote participant has been removed. Each `DdsDomainParticipant` that was able to communicate with it can no longer do so and must perform the following actions:
  - reset to `Unused` the involved `SoAdConnection`s
  - remove the association between the `SoConnId`s of the involved `SoAdConnection`s and the pair `<remote IP address, remote port>` of the removed Domain Participant

The explaned behaviour is formalized in [CP_SWS_Dds_01019], [CP_SWS_Dds_01020], [CP_SWS_Dds_01013], [CP_SWS_Dds_01014] and sequence diagrams in Chapter 9.3.1, Chapter 9.3.2.

7.2 General Requirements

Inside the same DDS Domain, topic names shall be unique according to "Section 2.2.1.2 Conceptual Outline" in [2].

[CP_SWS_Dds_CONSTR_00712] Topic name uniqueness ⌈During the validation of the configuration, a validation error shall be raised if different `DdsTopic`s within a DDS Domain share the same name.⌋

7.2.1 Communication requirements

7.2.1.1 Serialization requirements

[CP_SWS_Dds_00734] DDS Data serialization

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00006, FO_RS_Dds_00004, FO_RS_Dds_00019, FO_RS_Dds_00020

⌈The `Dds` shall perform the serialization of a `DdsAppDataTxPdu` into a `DdsRtpsDataTxPdu` or a `DdsRtpsMulticastDataTxPdu`. The PDU, which is used by the `DdsDataWriter` (either `DdsRtpsDataTxPdu` or `DdsRtpsMulticastDataTxPdu`), shall be selected at runtime according to the DDS middleware internal policies.⌋

[CP_SWS_Dds_00726] DDS-RTPS compliance

Upstream requirements: FO_RS_Dds_00002, FO_RS_Dds_00017, FO_RS_Dds_00019, FO_RS_Dds_00020

⌈Data produced into one of `DdsRtpsDataTxPdu`, `DdsRtpsDpDataTxPdu`, `DdsRtpsMulticastDataTxPdu`, `DdsRtpsDpMulticastDataTxPdu` shall be compliant with the DDS Wire Interoperability protocol (RTPS) defined in [8]. To guarantee such compliance, the `Dds` module shall serialize the payload according to the DDS standard serialization rules defined in section 7.4.3.5 of above mentioned document ([19]).⌋

[CP_SWS_Dds_00728] DDS serialization of primitive types

Upstream requirements: FO_RS_Dds_00004, FO_RS_Dds_00006, FO_RS_Dds_00007

⌈The Dds module shall serialize AUTOSAR primitive data types as described in [FO_PRS_DDS_00501], according to the standard serialization rules for the equivalent DDS PRIMITIVE_TYPE defined in section 7.2.2.2 of [19]⌋

Table 7.3 provides the equivalent DDS PRIMITIVE_TYPEs for the primitive AUTOSAR CP platform data types. For AUTOSAR CP platform data types, please refer to [20], chapter 8.2

Type	DDS Type	Remark
boolean	Boolean	
uint8	Byte	
uint16	UInt16	
uint32	UInt32	
uint64	UInt64	
sint8	Byte	
sint16	Int16	
sint32	Int32	
sint64	Int64	
uint8_least	Byte	
uint16_least	UInt16	
uint32_least	UInt32	
sint8_least	Byte	
sint16_least	Int16	
sint32_least	Int32	
float32	Float32	
float64	Float64	

Table 7.3: Serialization of primitive AUTOSAR CP platform data types

[CP_SWS_Dds_00729] DDS serialization of enumeration data types

Upstream requirements: FO_RS_Dds_00004, FO_RS_Dds_00006, FO_RS_Dds_00007

⌈The Dds module shall serialize ImplementationDataType of category ENUMERATION (refer to chapter 5.5.4 of [6]) as described in [FO_PRS_DDS_00502], according to the standard serialization rules for DDS ENUM_TYPE defined in section 7.2.2.4.1.1 of [19].⌋

[CP_SWS_Dds_00730] DDS serialization of ARRAY data type

Upstream requirements: FO_RS_Dds_00004, FO_RS_Dds_00006, FO_RS_Dds_00007

⌈The Dds module shall serialize ImplementationDataType of category ARRAY (refer to chapter 5.3.4.4 of [6]) as described in [FO_PRS_DDS_00507], according to the standard serialization rules for DDS ARRAY_TYPE defined in section 7.2.2.4.3 of [19].⌋

[CP_SWS_Dds_00731] DDS serialization of STRUCTURE data type

Upstream requirements: FO_RS_Dds_00004, FO_RS_Dds_00006, FO_RS_Dds_00007

⌈The Dds module shall serialize ImplementationDataType of category STRUCTURE (refer to chapter 7.2.2.4.4.1 of [6]) as described in [FO_PRS_DDS_00503], according to the standard serialization rules for DDS STRUCT_TYPE defined in section 7.4.3.5 of [19]. The Dds module shall mark as optional all optional members of the structure (refer to section 7.2.2.4.4.5 of [19].)⌋

[CP_SWS_Dds_CONSTR_00732] DDS serialization of UNION data type

⌈ImplementationDataType of category UNION (refer to chapter 7.2.2.4.4.2 of [6]) are not managed by the Dds.

The Dds configuration validation shall fail in case a DdsTopic links an ImplementationDataType which contains a union.⌋

[CP_SWS_Dds_CONSTR_00733] DDS serialization of POINTER data type

⌈ImplementationDataType of category POINTER (refer to chapter 7.2.2.4.6 of [6]) are not managed by the Dds.

The Dds configuration validation shall fail in case a DdsTopic links an ImplementationDataType which contains a pointer.⌋

[CP_SWS_Dds_00735] Encoding Format and Endianness of Strings in DDS

Upstream requirements: FO_RS_Dds_00004, FO_RS_Dds_00006, FO_RS_Dds_00007

⌈The Dds module shall encode Strings, as described in [FO_PRS_DDS_00504] and [FO_PRS_DDS_00505], according to Section 7.4.1.1.2 of [19].⌋

7.2.1.2 Deserialization requirements

[CP_SWS_Dds_00736] DDS Data deserialization

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00004, FO_RS_Dds_00006, FO_RS_Dds_00019, FO_RS_Dds_00020

⌈On receiving side, the Dds module shall deserialize a given DdsRtpsDataRxPdu according the DDS Wire Interoperability protocol ([8]) to an AUTOSAR compliant PDU. The DDS deserialized DdsRtpsDataRxPdu or DdsRtpsMulticastDataRxPdu is then provided as DdsAppDataRxPdu to upper layers.⌋

7.2.1.3 Transmission Queue management

The Dds module shall provide a Dds_Transmit function so that the PduR is able to initiate the transmission of a upper layer DdsAppDataTxPdu. When called, the Dds_Transmit function saves the received PDU into the queue and then returns.

For Dds_Transmit API specific requirements refer to the specific API Chapter 8.3.2.1.

[CP_SWS_Dds_00851] Internal transmission queues

Upstream requirements: FO_RS_Dds_00015

⌈The Dds module shall manage an internal set of queues where incoming transmission upper layer PDUs shall be stored.⌋

[CP_SWS_Dds_00828] Tx queues set processing order

Upstream requirements: FO_RS_Dds_00015

⌈The Transmission function (e.g., Dds_MainFunction_Tx shall establish the tx queues processing order based on the queue specific period. If some queue has the same period value, their mutual order is not defined.⌋

[CP_SWS_Dds_00838] Tx queue processing rules

Upstream requirements: FO_RS_Dds_00015

⌈For each queue the Transmission function shall perform all the DDS Middleware processing (QoS management, safety and security tasks, se-rialization) on received data, according to the configurated processing algorithm.⌋

The transmission queue shall have its own processing period. The queue specific period shall be a multiple of the DdsGeneral.DdsMainTxFunctionPeriod.

If more than one DdsAppDataTxPdu are stored in queue then transmission of next PDU from the Tx queue shall be done after the Dds_TxConfirmation for previous transmission is received.

[CP_SWS_Dds_00837] Tx queue processing algorithm

Upstream requirements: FO_RS_Dds_00015

⌈The order of the queue processing shall depend by the configurated DdsQueueAlgorithm:

- FIFO: the oldest received transmission upper layer PDU present in queue shall be processed first
- LIFO: the latest received transmission upper layer PDU present in queue shall be processed first
- VENDOR_SPECIFIC: Implementation/Vendor specific processing algorithm

⌋

7.2.1.4 Transmission requirements

The Dds module shall provide a Dds_MainFunction_Tx function to perform the actual DDS Middleware processing (and subsequent PDU transmission). For Dds_MainFunction_Tx specific requirements refer to the specific Chapter 8.4.2.

DDSI-RTPS [8] supports both UDP unicast and UDP multicast when sending RTPS messages. Also, they aren't mutually exclusive: just one or both can be used in the same domain according to the system's communications design constraints (network architecture, performance, safety, security, etc.). To support both unicast and multicast address, the SoAd shall be properly configured.

[CP_SWS_Dds_CONSTR_00865] Unicast transmission ⌈During validation of the configuration, a validation error shall be raised if a DdsRtpsDataTxPdu or a DdsRtpsDpDataTxPdu belong to a SoAdSocketConnection configured with a multicast SoAdSocketRemoteIpAddress.⌋

[CP_SWS_Dds_CONSTR_00866] Multicast transmission ⌈During validation of the configuration, a validation error shall be raised if a DdsRtpsMulticastDataTxPdu or a DdsRtpsDpMulticastDataTxPdu belong to a SoAdSocketConnection configured with a unicast SoAdSocketRemoteIpAddress.⌋

[CP_SWS_Dds_00720] DdsAppDataTxPdu Meta Data handling

Upstream requirements: SRS_BSW_00496

⌈The Dds shall ensure that MetaData carried by a DdsAppDataTxPdu will be present unchanged in the corresponding DdsRtpsDataTxPdu.⌋

[CP_SWS_Dds_00722] DdsRtpsDataTxPdu Meta Data handling in DdsDataSender context

Upstream requirements: SRS_BSW_00496

⌈If a DdsRtpsDataTxPdus meets the following conditions:

- it refers to a global PDU structure configured with a MetaDataItem of the type SOCKET_CONNECTION_ID_16
- it is the result of the processing of some DdsAppDataTxPdu referred by a DdsDataWriter referred by a DdsDataSender.DdsSenderDataWriterRef

then the Dds shall use PduInfoType.MetaDataPtr to provide the socket connection identifier (SoConId) of the SoAdSocketConnection suitable to reach the required remote Domain Participant.⌋

Note. MetaData is defined in [21, AUTOSAR_TPS_ECUConfiguration]

Note. Usage of MetaDataItem of the type SOCKET_CONNECTION_ID_16 is not mandatory, it depends on how the SoAd is configured (See Chapter F.1). Requirement [CP_SWS_Dds_00722] states that Dds shall support this feature

[CP_SWS_Dds_00737] RTPS transmission PDUs for discovery purpose

Upstream requirements: FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈In "All dynamic" scenario or in "Partially dynamic" scenario, the Dds shall use at least one of the two types of PDUs: DdsRtpsDpDataTxPdu or DdsRtpsDpMulticastDataTxPdu for discovery transmission purpose.⌋

7.2.1.4.1 Application I-PDU immediate Tx confirmation

The point in time at which the Dds module confirms the transmission of a DdsAppDataTxPdu to the upper layer module is determined by the parameter DdsAppDataTxPdu.DdsImmediateTxConfirmation according to the following [SWS_Dds_10001] requirement:

[SWS_Dds_10001] Application PDU immediate transmission confirmation to the upper layer

Upstream requirements: FO_RS_Dds_00015

⌈If DdsAppDataTxPdu.DdsImmediateTxConfirmation is set to TRUE then the Dds module shall call PduR_DdsTxConfirmation for the DdsAppDataTxPdu with result E_OK once the DdsAppDataTxPdu has been copied to the DdsTxQueue, regardless of the transmission of the corresponding DdsRtpsDataTxPdus, DdsRtpsMulticastDataTxPdus on the bus.⌋

7.2.1.5 Reception Queue management

[CP_SWS_Dds_00864] Internal reception queues

Upstream requirements: FO_RS_Dds_00016

⌈The Dds module shall manage an internal set of queues where incoming reception lower layer PDUs shall be stored.⌋

[CP_SWS_Dds_00825] Rx queues set processing order

Upstream requirements: FO_RS_Dds_00016

⌈The reception function (e.g., Dds_MainFunction_Rx) shall establish the Rx queues processing order based on the queue period. If some queue has the same DdsGeneral.DdsMainRxFunctionPeriod value, their mutual order is not defined.⌋

[CP_SWS_Dds_00834] Rx queue processing rules

Upstream requirements: FO_RS_Dds_00016

⌈For each queue the reception function shall perform all the DDS Middleware operations (QoS management, safety and security tasks, de-serialization) on received data, according to the configurated processing algorithm.⌋

The reception queue shall have its own processing period. The queue specific period shall be a multiple of the DdsGeneral.DdsMainRxFunctionPeriod.

[CP_SWS_Dds_00836] Rx queue processing algorithm

Upstream requirements: FO_RS_Dds_00016

⌈The order of the queue processing shall depend by the configurated algorithm:

- FIFO: the oldest received lower layer PDU present in queue shall be processed first
- LIFO: the latest received lower layer PDU present in queue shall be processed first
- VENDOR_SPECIFIC: the order of processing is not specified

⌋

7.2.1.6 Reception requirements

Every reception lower layer PDU which is received by the Ethernet Interface is given to the PDU Router by means of the SoAd. The PDU Router routes those PDUs to the Dds reception interface invoking the Dds_RxIndication callback.

For Dds_RxIndication API specific requirements refer to the specific API Chapter 8.3.2.2.

Similar to transmission, DDSI-RTPS [8] supports both UDP unicast and UDP multicast also when receiving RTPS messages. Also, they aren't mutually exclusive: just one or both can be used in the same domain according to the system's communications design constraints (network architecture, performance, safety, security, etc.).

To support both unicast and multicast address, the SoAd shall be properly configured.

[CP_SWS_Dds_CONSTR_00867] Unicast reception ⌈During validation of the configuration, a validation error shall be raised if a DdsRtpsDataRxPdu or a DdsRtpsDpDataRxPdu belong to a SoAdSocketConnection configured with a multicast SoAdSocketRemoteIpAddress.⌋

[CP_SWS_Dds_CONSTR_00868] Multicast reception ⌈During validation of the configuration, a validation error shall be raised if a DdsRtpsMulticastDataRxPdu or a DdsRtpsDpMulticastDataRxPdu belong to a SoAdSocketConnection configured with a unicast SoAdSocketRemoteIpAddress.⌋

[CP_SWS_Dds_00721] DdsRtpsDataRxPdu Meta Data handling

Upstream requirements: SRS_BSW_00496

⌈The Dds shall ensure that MetaData carried by a DdsRtpsDataRxPdu will be present unchanged in the corresponding DdsAppDataRxPdu.⌋

[CP_SWS_Dds_00738] RTPS reception PDUs for discovery purpose

Upstream requirements: FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈In "All dynamic" scenario or in "Partially dynamic" scenario, the Dds shall use at least one of the two types of PDUs: DdsRtpsDpDataRxPdu or DdsRtpsDpMulticastDataRxPdu for discovery reception purpose.⌋

7.2.1.7 Timing requirements

[CP_SWS_Dds_00873] Processing timestamp

Upstream requirements: FO_RS_Dds_00015, FO_RS_Dds_00016

⌈If needed, the DDS middleware shall obtain the timestamp by invoking the `StbM_GetCurrentTime` API. The `Dds` module shall call the API `Stbm_GetCurrentTime` with the configured time base reference (see DdsGeneral.DdsSynchronizedTimeBaseRef) to get the Timestamp needed for the RTPS packet.⌋

[CP_SWS_Dds_00859] RTPS Timestamp

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00015

⌈The Timestamp needed for the RTPS packet shall be provided by the `StbM_GetCurrentTime` StbM API. The `Dds` module shall call the API `Stbm_GetCurrentTime` with the configured time base reference (see DdsGeneral.DdsSynchronizedTimeBaseRef) to get the Timestamp needed for the RTPS packet.⌋

Note: The AUTOSAR `Dds` module is not responsible to time stamp received or transmitted PDUs. The responsibility is bound to the DDS middleware.

7.2.1.8 Service Oriented Architecture requirements

[CP_SWS_Dds_CONSTR_00960] DdsDataWriter employed in a service interface element ⌈If a certain `DdsDataWriter` is referenced by one of the following ECUC parameter:

- `DdsServiceMethodsRequestReplyMethod.DdsDataWriterRef`
- `DdsServiceFieldsFieldServerGetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldServerSetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldClientGetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldClientSetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldNotifierSender.DdsDataWriterRef`
- `DdsServiceMethodsFireAndForgetMethod.DdsDataWriterRef`
- `DdsServiceEventSender.DdsDataWriterRef`

then the upper multiplicity of `DdsDataWriter.DdsDataWriterAppTxPduRef` shall be 1.⌋

[CP_SWS_Dds_CONSTR_00961] DdsDataReader employed in a service interface element ⌈If a certain `DdsDataReader` is referenced by one of the following ECUC parameter:

- DdsServiceMethodsRequestReplyMethod.DdsDataReaderRef
- DdsServiceFieldsFieldServerGetter.DdsDataReaderRef
- DdsServiceFieldsFieldServerSetter.DdsDataReaderRef
- DdsServiceFieldsFieldClientGetter.DdsDataReaderRef
- DdsServiceFieldsFieldClientSetter.DdsDataReaderRef
- DdsServiceFieldsFieldNotifierReceiver.DdsDataReaderRef
- DdsServiceMethodsFireAndForgetMethod.DdsDataReaderRef
- DdsServiceEventReceiver.DdsDataReaderRef

then the upper multiplicity of DdsDataReader.DdsDataReaderAppRxPduRef shall be 1.⌋

7.2.1.8.1 Bidirectional service interface elements common requirements

[CP_SWS_Dds_00921] DdsAppDataRxPdu format in bidirectional service interface elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031

⌈The format of a certain DdsAppDataRxPdu referenced by one of:

- DdsServiceMethodsRequestReplyMethod.DdsDataReaderRef
- DdsServiceFieldsFieldServerGetter.DdsDataReaderRef
- DdsServiceFieldsFieldServerSetter.DdsDataReaderRef
- DdsServiceFieldsFieldClientGetter.DdsDataReaderRef
- DdsServiceFieldsFieldClientSetter.DdsDataReaderRef

shall be defined according to [CP_SWS_DdsXf_00143], [CP_SWS_DdsXf_00169].⌋

Note. The reason for [CP_SWS_Dds_00921] lies in the fact that the reception PDU produced by Dds is reasonably expected to be processed by the DDS Transformer, which expects a well-defined layout. Conversely, when the DDS Transformer must produce a PDU for transmission, it is Dds that assumes it is built with the same layout (See Chapter 7.1.2.2).

[CP_SWS_Dds_00925] DdsAppDataTxPdu requestId mapping on server side in bidirectional service interface elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00033

⌈Given a certain DdsAppDataTxPdu referenced by one of:

- DdsServiceMethodsRequestReplyMethod.DdsDataWriterRef

- DdsServiceFieldsFieldServerGetter.DdsDataWriterRef
- DdsServiceFieldsFieldServerSetter.DdsDataWriterRef

the requestId got from DdsAppDataTxPdu shall be mapped as follows:

- ReplyHeader.relatedRequestId.writer_guid.entityKey = requestId.clientId
- ReplyHeader.sequenceCounter = requestId.sequenceCounter

ReplyHeader is the standard Reply Header defined in [14, section 7.5.1.1.1 of DDS-RPC specification]⌋

[CP_SWS_Dds_00926] DdsAppDataTxPdu requestId mapping on client side in bidirectional service interface elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00033

⌈Given a certain DdsAppDataTxPdu referenced by one of:

- DdsServiceMethodsRequestReplyMethod.DdsDataWriterRef
- DdsServiceFieldsFieldClientGetter.DdsDataWriterRef
- DdsServiceFieldsFieldClientSetter.DdsDataWriterRef

the requestId got from DdsAppDataTxPdu shall be mapped as follows:

- RequestHeader.requestId.writer_guid.entityKey = requestId.clientId
- RequestHeader.sequenceCounter = requestId.sequenceCounter

RequestHeader is the standard Request Header defined in [14, section 7.5.1.1.1 of DDS-RPC specification]⌋

[CP_SWS_Dds_00927] DdsAppDataRxPdu requestId mapping on server side in bidirectional service interface elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00033

⌈Given a certain DdsAppDataRxPdu referenced by one of:

- DdsServiceMethodsRequestReplyMethod.DdsDataReaderRef
- DdsServiceFieldsFieldServerGetter.DdsDataReaderRef
- DdsServiceFieldsFieldServerSetter.DdsDataReaderRef

the requestId shall be retrieved as follows:

- requestId.clientId = RequestHeader.requestId.writer_guid.entityKey
- requestId.sequenceCounter = RequestHeader.sequenceCounter

`RequestHeader` is the standard Request Header defined in [14, section 7.5.1.1.1 of DDS-RPC specification]⌋

[CP_SWS_Dds_00928] DdsAppDataRxPdu requestId mapping on client side in bidirectional service interface elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00033

⌈Given a certain `DdsAppDataRxPdu` referenced by one of:

- `DdsServiceMethodsRequestReplyMethod.DdsDataReaderRef`
- `DdsServiceFieldsFieldClientGetter.DdsDataReaderRef`
- `DdsServiceFieldsFieldClientSetter.DdsDataReaderRef`

the `requestId` shall be retrieved as follows:

- `requestId.clientId = ReplyHeader.relatedRequestId.writer_guid.entityKey`
- `requestId.sequenceCounter = ReplyHeader.sequenceCounter`

`ReplyHeader` is the standard Reply Header defined in [14, section 7.5.1.1.1 of DDS-RPC specification]⌋

In accordance with the DDS-RPC [14] specification, the return code generated by the application server (hereinafter referred to as `returnValue`) shall be mapped into the reply message sent to the client, based on the corresponding [14, DDS-RPC IDL template] that defines its structure and semantics. Since this error code is represented as an enumerated type and the value of the `returnValue` field varies depending on the specific application, current specification does not establish any general rule regarding such mapping.

This document restricts itself to prescribing the exact mapping of the return code, in order to ensure full compatibility with an interface compliant with **AUTOSAR Adaptive Platform** specifications (See following [CP_SWS_Dds_00930], [CP_SWS_Dds_00931]).

[CP_SWS_Dds_00930] returnValue mapping on server side in bidirectional service elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00033

⌈If a certain `DdsAppDataTxPdu` referenced by one of:

- `DdsServiceMethodsRequestReplyMethod.DdsDataWriterRef`
- `DdsServiceFieldsFieldServerGetter.DdsDataWriterRef`
- `DdsServiceFieldsFieldServerSetter.DdsDataWriterRef`

is defined according to [CP_SWS_DdsXf_00143], [CP_SWS_DdsXf_00169], then,

the returnValue shall be mapped to the ErrorCode field defined according to [FO_PRS_DDS_00303].

⌋

[CP_SWS_Dds_00931] returnValue mapping on client side in bidirectional service elements context

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00033

⌈If a certain DdsAppDataRxPdu referenced by one of:

- DdsServiceMethodsRequestReplyMethod.DdsDataReaderRef
- DdsServiceFieldsFieldClientGetter.DdsDataReaderRef
- DdsServiceFieldsFieldClientSetter.DdsDataReaderRef

is defined according to [CP_SWS_Dds_00921],

the returnValue shall be copied by ErrorCode field defined according to [FO_PRS_DDS_00303].

⌋

7.2.1.8.2 Unidirectional service interface elements common requirements

[CP_SWS_Dds_00723] DdsRtpsDataTxPdu Meta Data handling in unidirectional service elements context

Upstream requirements: SRS_BSW_00496

⌈If a DdsRtpsDataTxPdus meets the following conditions:

- it refers to a global PDU structure configured with a MetaDataItem of the type SOCKET_CONNECTION_ID_16
- it is the result of the processing of some DdsAppDataTxPdu referred by one of the following:
  - DdsServiceEventSender.DdsDataWriterRef
  - DdsServiceMethodsFireAndForgetMethod.DdsDataWriterRef on client side
  - DdsServiceFieldsFieldNotifierSender.DdsDataWriterRef

then the Dds shall use PduInfoType.MetaDataPtr to provide the socket connection identifier (SoConId) of the socket connection suitable to reach the required remote Domain Participant.⌋

Note. MetaData is defined in [21, AUTOSAR_TPS_ECUConfiguration]

7.2.1.8.3 Events

[CP_SWS_Dds_00900] Configuration and manipulation of DdsTopic referenced by a DdsDataWriter referenced by a DdsServiceEventSender object

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00030

⌈Sending of Event samples through the DdsTopic referenced by DdsDataWriter referenced by a DdsServiceEventSender shall be implemented according to [FO_PRS_DDS_00100], [FO_PRS_DDS_00101] and [FO_PRS_DDS_00102] of [15].⌋

[CP_SWS_Dds_00902] Configuration and manipulation of DdsTopic referenced by a DdsDataReader referenced by a DdsServiceEventReceiver object

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00030

⌈Receiving of Event samples through the DdsTopic referenced by DdsDataReader referenced by a DdsServiceEventReceiver shall be implemented according to [FO_PRS_DDS_00100], [FO_PRS_DDS_00101], [FO_PRS_DDS_00103], [FO_PRS_DDS_00104] and [FO_PRS_DDS_00108] of [15].⌋

7.2.1.8.4 Methods

7.2.1.8.4.1 Request and response Methods

[CP_SWS_Dds_00904] Configuration and manipulation of DdsTopics referenced in a Request and Response Method context on client side

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031

⌈In a certain DdsConsumedServiceInstance, calling of Methods defined by DdsServiceMethodsRequestReplyMethod through the DdsTopics referenced by a DdsDataWriter or by a DdsDataReader,

shall be implemented according to [FO_PRS_DDS_00300], [FO_PRS_DDS_00301], [FO_PRS_DDS_00302], [FO_PRS_DDS_00303], [FO_PRS_DDS_00304], [FO_PRS_DDS_00305], [FO_PRS_DDS_00308] and [FO_PRS_DDS_00309] of [15].

The DdsTopicImplementationType referenced by the DdsTopic referenced by the DdsDataReader and DdsDataWriter shall be ignored.⌋

[CP_SWS_Dds_00905] Configuration and manipulation of DdsTopics referenced in a Request and Response Method context on server side

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031, FO_RS_Dds_00032

⌈In a certain DdsProvidedServiceInstance, dispatching of Methods defined by DdsServiceMethodsRequestReplyMethod through the DdsTopics referenced by a DdsDataWriter or by a DdsDataReader,

shall be implemented according to [FO_PRS_DDS_00300], [FO_PRS_DDS_00301], [FO_PRS_DDS_00302], [FO_PRS_DDS_00303], [FO_PRS_DDS_00306],

[FO_PRS_DDS_00307], [FO_PRS_DDS_00309], [FO_PRS_DDS_00312] and [FO_PRS_DDS_00313] of [15].

The DdsTopicImplementationType referenced by the DdsTopic referenced by the DdsDataReader and DdsDataWriter shall be ignored.⌋

7.2.1.8.4.2 Fire and Forget Methods with Arguments

[CP_SWS_Dds_00906] Handling of "Fire and Forget" methods with arguments

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031

⌈"Fire and Forget" or "responseless" methods defined as DdsServiceMethodsFireAndForgetMethod with parameters (i.e. referencing a VARIABLE-DATA-PROTOTYPE within a SenderReceiverInterface) shall be handled according to the rules described in [CP_SWS_Dds_00905] and [CP_SWS_Dds_00906], excluding all mechanisms related to preparation, transmission, reception and processing of the method's reply.⌋

7.2.1.8.4.3 Fire and Forget Methods without Arguments

[CP_SWS_Dds_00907] Handling of "Fire and Forget" methods without arguments

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00031

⌈"Fire and Forget" or "responseless" methods defined as DdsServiceMethodsFireAndForgetMethod without parameters (i.e. referencing a TRIGGER within a TriggerInterface) shall be handled according to the rules described in [CP_SWS_Dds_00905] and [CP_SWS_Dds_00906], excluding all mechanisms related to preparation, transmission, reception and processing of the method's reply.⌋

[CP_SWS_Dds_CONSTR_00956] Fire and Forget Method DdsDataElementRef ⌈if DdsServiceMethodsFireAndForgetMethod.DdsDataElementRef is defined then DdsServiceMethodsFireAndForgetMethod.DdsTriggerRef shall not.⌋

[CP_SWS_Dds_CONSTR_00957] Fire and Forget Method DdsTriggerRef ⌈if DdsServiceMethodsFireAndForgetMethod.DdsTriggerRef is defined then DdsServiceMethodsFireAndForgetMethod.DdsDataElementRef shall not.⌋

7.2.1.8.5 Fields

7.2.1.8.5.1 Notifier

[CP_SWS_Dds_00901] Configuration and manipulation of DdsTopic referenced by a DdsDataWriter in a Field Notifier context on server side

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00033

⌈Sending of Field Notifier samples through the `DdsTopic` referenced by `DdsDataWriter` referenced by a `DdsServiceFieldsFieldNotifierSender`

shall be implemented according to [FO_PRS_DDS_00400], [FO_PRS_DDS_00401] and [FO_PRS_DDS_00402] of [15].⌋

[CP_SWS_Dds_00903] Configuration and manipulation of DdsTopic referenced by a DdsDataReader in a Field Notifier context on client side

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00033

⌈Receiving of Field Notifier samples through the `DdsTopic` referenced by `DdsDataReader` referenced by a `DdsServiceFieldsFieldNotifierReceiver`

shall be implemented according to [FO_PRS_DDS_00400], [FO_PRS_DDS_00401], [FO_PRS_DDS_00403] and [FO_PRS_DDS_00404] of [15].⌋

7.2.1.8.5.2 Getter and Setter

[CP_SWS_Dds_00908] Configuration and manipulation of DdsTopics referenced by a DdsDataWriter or by a DdsDataReader, in a Fields Getter/Setter context on client side

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00033, FO_RS_Dds_00032

⌈In a certain `DdsConsumedServiceInstance`, calling of Field Get/Set Methods through the `DdsTopic`s referenced by a `DdsDataWriter` or by a `DdsDataReader`, referenced by a `DdsServiceFieldsFieldClientGetter` or by `DdsServiceFieldsFieldClientSetter`,

shall be implemented according to [FO_PRS_DDS_00407], [FO_PRS_DDS_00408],[FO_PRS_DDS_00409], [FO_PRS_DDS_00410], [FO_PRS_DDS_00411], [FO_PRS_DDS_00414] and [FO_PRS_DDS_00415] of [15].

The `DdsTopicImplementationType` referenced by the `DdsTopic` referenced by the `DdsDataReader` and `DdsDataWriter` shall be ignored.⌋

[CP_SWS_Dds_00909] Configuration and manipulation of DdsTopics referenced by a DdsDataWriter or by a DdsDataReader, in a Fields Getter/Setter context on server side

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00033, FO_RS_Dds_00032

⌈In a certain DdsProvidedServiceInstance, dispatching of Field Get/Set Methods through the DdsTopics referenced by a DdsDataWriter or by a DdsDataReader, referenced by a DdsServiceFieldsFieldClientGetter or by DdsServiceFieldsFieldClientSetter,

shall be implemented according to [FO_PRS_DDS_00407], [FO_PRS_DDS_00408],[FO_PRS_DDS_00409], [FO_PRS_DDS_00412], [FO_PRS_DDS_00413], [FO_PRS_DDS_00415], [FO_PRS_DDS_00418] and [FO_PRS_DDS_00419] of [15].

The DdsTopicImplementationType referenced by the DdsTopic referenced by the DdsDataReader and DdsDataWriter shall be ignored.⌋

7.2.1.9 Dynamic discovery requirements

Requirements for protocols support

[CP_SWS_Dds_01010] SPDP support

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040

⌈If a DdsDomainParticipant does not define any DdsRemoteDomainParticipantCollection.DdsRemoteDomainParticipant within its DdsRemoteDomainParticipantCollection, then it shall employ **SPDP** according to [8, DDS-RTPS at Chapter 8.5].⌋

[CP_SWS_Dds_01011] SEDP support

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00041

⌈If a DdsDomainParticipant defines a DdsRemoteDomainParticipantCollection.DdsRemoteDomainParticipant within its DdsRemoteDomainParticipantCollection and no DdsRemoteDomainParticipant.DdsRemoteDataReader or DdsRemoteDomainParticipant.DdsRemoteDataWriter, then it shall employ **SEDP** according to [8, DDS-RTPS at Chapter 8.5].⌋

[CP_SWS_Dds_01012] Dynamic Service Instance Discovery support

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈If DdsConsumedServiceInstance.DdsRequiredServiceInstanceId is set to **ANY**, then the Dds module shall employ **Dynamic Service Instance Discovery** according to [18, FO-PRS-DDSServiceDiscoveryProtocol].⌋

[CP_SWS_Dds_01015] Discovery behaviour on network fail

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈If the transmission of one of DdsRtpsDpDataTxPdu or DdsRtpsDpMulticastDataTxPdu fails, Dds shall continue retransmitting that PDU until it completes successfully.⌋

[CP_SWS_Dds_01031] Remote participant connection access

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈For each DdsDomainParticipant, Dds shall optionally access a generic connection by means of:

- DdsRtpsDataTxPdu.DdsSoAdSocketConnectionRef for unicast traffic
- DdsRtpsMulticastDataTxPdu.DdsSoAdSocketConnectionRef for multicast traffic
- DdsRtpsDpDataTxPdu.DdsSoAdSocketConnectionRef for unicast discovery traffic
- DdsRtpsDpMulticastDataTxPdu.DdsSoAdSocketConnectionRef for multicast discovery traffic

⌋

[CP_SWS_Dds_01013] Dds actions after a remote Domain Participant discover event

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈When a remote participant is discovered, the Dds shall configure, within a SoAdConnection, the necessary <IP address, port> pair for sending messages to those participant.⌋

[CP_SWS_Dds_01014] Dds actions after a remote Domain Participant removal event

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈When a remote participant is removed, the Dds shall mark as unused the SoAdConnection configured with the <IP address, port> pair required for sending messages to those participant.⌋

[CP_SWS_Dds_01019] Setting remote Domain Participant address

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈To satisfy [CP_SWS_Dds_01013], the Dds shall use one of the following APIs:

- SoAd_SetUniqueRemoteAddr: according to Chapter 8.5.2, see 8.3.4.15 [5]

- SoAd_SetRemoteAddr: according to Chapter 8.5.2, see 8.3.4.14 [5]

⌋

[CP_SWS_Dds_01020] Resetting remote Domain Participant address

Upstream requirements: FO_RS_Dds_00001, FO_RS_Dds_00040, FO_RS_Dds_00041, FO_RS_Dds_00042, FO_RS_Dds_00043

⌈To satisfy [CP_SWS_Dds_01014], the Dds shall use one of the following APIs:

- SoAd_ReleaseRemoteAddr according to Chapter 8.5.2, see 8.3.4.15 [5]

⌋

7.2.1.10 Asynchronous copying to local buffer

Dds module supports asynchronous copying of a PDU into its local buffer (e.g. DMA-copy). This behavior is controlled per PDU by the ECU Configuration parameter KeepLocalPduBuffer. If KeepLocalPduBuffer is set to FALSE, the copying process is performed synchronously in the context of the transmission or reception api call (e.g. CPU-copy). If KeepLocalPduBuffer is set to TRUE, the source module keeps its buffer until Dds indicates that the buffer may be released and, depending on EcucDmaThreshold, the copy may be performed asynchronously.

For transmission this concerns the Dds application I-PDU provided by the upper layer (e.g: the I-PDU referenced by the DdsAppDataTxPdu) and Discovery RTPS I-PDU DdsRtpsDpMulticastDataTxPdu provided locally by the Dds module; for reception this concerns the Dds RTPS I-PDU received from the lower layer (e.g: the I-PDU referenced by the DdsRtpsDataRxPdu, DdsRtpsMulticastDataRxPdu, DdsRtpsDpDataRxPdu, DdsRtpsDpMulticastDataRxPdu).

The release procedures for the involved buffers differ when considering the two distinct processing paths. The invocation of PduR_DdsTxConfirmation by the Dds notifies the upper module that the buffer associated with the application I-PDU may be released, whereas the invocation of the PduR_DdsReleaseBuffer toward the lower module signifies the request to free the buffer of the received RTPS I-PDU. Finally, the upper module is required to invoke the Dds_ReleaseRxBuffer API, thereby enabling the Dds to release the buffer of the received application I-PDU.

[SWS_Dds_10002] Asynchronous copying of Tx application I-PDU to local buffer

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the Dds application I-PDU is set to TRUE and its SduLength is greater than EcucDmaThreshold, then the Dds module shall prepare to asynchronously copy the Dds application I-PDU into the DdsTxQueue and shall return from Dds_Transmit without waiting for the copy to complete.⌋

[SWS_Dds_10003] Synchronous copying of Tx application I-PDU to local buffer

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the Dds application I-PDU is set to TRUE and its SduLength does not exceed EcucDmaThreshold, then the Dds module shall copy the Dds application I-PDU into the DdsTxQueue synchronously within Dds_Transmit.⌋

[SWS_Dds_10004] Tx application I-PDU processing after successful asynchronous copy to local buffer

Upstream requirements: FO_RS_Dds_00015

⌈As soon as an asynchronous copying for Dds application I-PDU into the DdsTxQueue has been successfully finalized, the Dds module shall process the data in a sub sequential call of Dds_MainFunction_Tx.⌋

[SWS_Dds_10005] Copying of Tx application I-PDU (KeepLocalPduBuffer set to TRUE) to local buffer fails

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the Dds application I-PDU is set to TRUE and the copying of the Dds application I-PDU into the DdsTxQueue fails, the Dds module shall call PduR_DdsTxConfirmation to indicate that the source module may release its buffer, shall free its DdsTxQueue buffer, and shall report the runtime error DDS_E_SAMPLE_LOST to the Det module.⌋

[SWS_Dds_10006] Asynchronous copying of Rx RTPS I-PDU to local buffer

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the received Dds RTPS I-PDU is set to TRUE and its SduLength is greater than EcucDmaThreshold, the Dds module shall shall prepare to asynchronously copy the DdsRtpsDataRxPdu, DdsRtpsMulticastDataRxPdu, DdsRtpsDpDataRxPdu, DdsRtpsDpMulticastDataRxPdu into the DdsRxQueue and shall return from Dds_RxIndication without waiting for the copy to complete.⌋

[SWS_Dds_10007] Synchronous copying of Rx RTPS I-PDU to local buffer

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the received Dds RTPS I-PDU is set to TRUE and its SduLength is not greater than EcucDmaThreshold, the Dds module shall copy the DdsRtpsDataRxPdu, DdsRtpsMulticastDataRxPdu, DdsRtpsDpDataRxPdu, DdsRtpsDpMulticastDataRxPdu into the DdsRxQueue synchronously within Dds_RxIndication.⌋

[SWS_Dds_10008] Rx RTPS I-PDU processing after successful asynchronous copy to local buffer

Upstream requirements: FO_RS_Dds_00015

⌈As soon as an asynchronous copying for RTPS I-PDU into the DdsRxQueue has been successfully finalized, the Dds module shall process the data in a sub sequential call of Dds_MainFunction_Rx.⌋

[SWS_Dds_10009] Rx RTPS I-PDU buffer release after local buffer copy

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the received Dds RTPS I-PDU is set to TRUE, the Dds module shall call PduR_DdsReleaseRxBuffer for that Dds RTPS I-PDU in Dds_MainFunction_Rx) once the copy of the Dds RTPS I-PDU into the DdsRxQueue has been finalized.⌋

[SWS_Dds_10010] Rx RTPS I-PDU buffer copy failure

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the Dds application I-PDU is set to TRUE and if the copy of the received Dds RTPS I-PDU into the the DdsRxQueue fails, the Dds module shall call PduR_DdsReleaseRxBuffer to indicate that the source module may release its buffer, shall free its related DdsRxQueue buffer, and shall report the runtime error DDS_E_SAMPLE_LOST to the Det module.⌋

[SWS_Dds_10011] RTPS I-PDU memory alignment for transmission

Upstream requirements: FO_RS_Dds_00015

⌈If KeepLocalPduBuffer of the Dds RTPS I-PDU is set to TRUE and this RTPS I-PDU needs to be transmitted, then the provided data buffer shall respect the memory alignment value PduBufferAlignmentof the referenced Pdu.⌋

Note: KeepLocalPduBuffer set to TRUE means that an I-PDU could potentially asynchronously be copied by the lower layer modules. The memory alignment would support a proper processing when data is added to the hardware buffer.

7.2.2 Security requirements

[CP_SWS_Dds_00750] DDS-security

Upstream requirements: FO_RS_Dds_00009

⌈In order to be compliant and to intercommunicate with other DDS systems, the Dds module shall implement security mechanisms by using DDS-Security Specification [16].⌋

[CP_SWS_Dds_00752] MAC usage

Upstream requirements: FO_RS_Dds_00009

⌈The Dds module shall guarantee data-integrity and message authentication at DomainParticipant level by adding a Message Authentication Code (MAC) to the message to be sent, calculated by using symmetric key algorithms. The resulting message shall still be DDSI-RTPS compliant.⌋

[CP_SWS_Dds_00753] CSM library usage

Upstream requirements: FO_RS_Dds_00009

⌈The Dds shall configure, for each DomainParticipant, one reference to each CSM job needed: one job to calculate MAC (DdsDomainParticipantCsmAuthenticateJob) and one to check MAC of received messages (DdsDomainParticipantCsmVerifyJob). For configuration details, refer to DdsDomainParticipantCryptoInfo. At sender side, the Dds shall add the resulting MAC of DdsDomainParticipantCsmAuthenticateJob to each message of this DomainParticipant. At receiving side, the Dds shall check the result of the DdsDomainParticipantCsmVerifyJob.⌋

[CP_SWS_Dds_CONSTR_00754] CSM job configuration ⌈The CSM DdsDomainParticipantCsmAuthenticateJob shall be configured to call Csm_MacGenerate and the DdsDomainParticipantCsmVerifyJob to call Csm_MacVerify.

The Dds configuration validation shall fail in case the DdsDomainParticipantCsmAuthenticateJob/DdsDomainParticipantCsmVerifyJob related to the same DdsDomainParticipant link CSM jobs that are not configured with Csm_MacGenerate and Csm_MacVerify respectively.

For configuration details, refer to DdsDomainParticipantCryptoInfo.⌋

[CP_SWS_Dds_CONSTR_00743] CSM key configuration ⌈Each CSM authenticate/verify pair, related to a single DomainParticipant, shall use the same keys (only simmetric-key algorithms are supported). For each DomainParticipant, the CSM used jobs shall be configured with the same keys.

The Dds configuration validation shall fail in case the DdsDomainParticipantCsmAuthenticateJob/DdsDomainParticipantCsmVerifyJob related to the same DdsDomainParticipant link CSM jobs that are not configured with the same key.

For configuration details, refer to DdsDomainParticipantCryptoInfo.⌋

[CP_SWS_Dds_00756] MAC calculation failure

Upstream requirements: FO_RS_Dds_00009

⌈If the MAC calculation fails (e.g., the Csm_MacGenerate or Csm_MacVerify return any error), the Dds module shall call the API Det_ReportRuntimeError with the DDS_E_CSM_LIBRARY_ERROR runtime error code and discard the message to be sent.

In this case, during transmission the Dds shall call the PduR_DdsTxConfirmation function with result = E_NOT_OK.⌋

[CP_SWS_Dds_00758] MAC check failure

Upstream requirements: FO_RS_Dds_00009

⌈At receiving side, if the MAC check fails, the Dds module shall call the API Det_ReportRuntimeError with the DDS_E_CSM_CHECK_FAILED rutime error code and discard the message.⌋

7.2.3 Safety requirements

[CP_SWS_Dds_00761] Repetition or Insertion of Information

Upstream requirements: FO_RS_Dds_00010

⌈The Dds module shall, as described in [FO_PRS_DDS_00601], use submessages which have counters, e.g., AckNack, Data and DataFrag, etc., to guarantee safety mechanisms against Repetition or Insertion of Information faults. At receiving side, if a message with a duplicated counter is received, the Dds module shall discard the message and call the API Det_ReportRuntimeError with the DDS_E_SAMPLE_REJECTED.⌋

[CP_SWS_Dds_00762] Loss or Incorrect sequence of Information

Upstream requirements: FO_RS_Dds_00010

⌈The Dds module shall, as described in [FO_PRS_DDS_00602], use submessages which have counters, e.g., AckNack, Data and DataFrag, etc., to guarantee safety mechanisms against Loss or Incorrect sequence of Information faults. At receiving side, if a message with a non-consecutive counter is received, the Dds module shall discard the message and call the API Det_ReportRuntimeError with the DDS_E_SAMPLE_LOST runtime error code.⌋

The Dds module shall use QoSs able to monitor timeouts, such as DEADLINE, LATENCY_BUDGET, LIFESPAN and TIME_BASED_FILTER (refer respectively to DdsDeadline, DdsLatencyBudget, DdsLifespan and DdsTimeBasedFilter) to guarantee safety mechanisms against Delay of Information fault (take [2] for details on those QoS policies).

[CP_SWS_Dds_00763] Delay of Information - sending checks

Upstream requirements: FO_RS_Dds_00005, FO_RS_Dds_00010

⌈At sending side, if some timing constraint is not fulfilled, the Dds module shall, as described in [FO_PRS_DDS_00603], discard the message and call the API Det_ReportRuntimeError with the DDS_E_SENDER_TIMING_MISSED runtime error code.⌋

[CP_SWS_Dds_00764] Delay of Information - receiving checks

Upstream requirements: FO_RS_Dds_00005, FO_RS_Dds_00010

⌈At receiving side, if some timing constraint is not fulfilled, the Dds module shall, as described in [FO_PRS_DDS_00603], discard the message and call the API Det_ReportRuntimeError with the DDS_E_RECEIVER_TIMING_MISSED runtime error code.⌋

[CP_SWS_Dds_00766] Corruption of Information

Upstream requirements: FO_RS_Dds_00010

⌈The Dds module shall, as described in [FO_PRS_DDS_00604], use CRC check to guarantee safety mechanisms against Corruption of Information fault.⌋

[CP_SWS_Dds_00769] CRC check failure

Upstream requirements: FO_RS_Dds_00010

⌈On received side, if the CRC check fails, the `Dds` module shall, as described in [FO_PRS_DDS_00604], call the API `Det_ReportRuntimeError` with the `DDS_E_CRC_CHECK_FAILED` runtime error code and discard the message.⌋

7.3 Error Classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" [4] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.3.1 Development Errors

[CP_SWS_Dds_00772] Definition of development errors in module Dds

Upstream requirements: FO_RS_Dds_00015, FO_RS_Dds_00016

⌈

Type of error	*Related error code*	*Error value*
Module not initialized	DDS_E_UNINIT	0x00
Null pointer has been passed as an argument	DDS_E_PARAM_POINTER	0x02
Invalid Upper Layer PduId	DDS_E_U_PDUID_INVALID	0x03
Invalid Lower Layer PduId	DDS_E_L_PDUID_INVALID	0x04

⌋

7.3.2 Runtime Errors

[CP_SWS_Dds_00773] Definition of runtime errors in module Dds

Upstream requirements: FO_RS_Dds_00005, FO_RS_Dds_00009, FO_RS_Dds_00010, FO_RS_Dds_00015, FO_RS_Dds_00016

⌈

Type of error	*Related error code*	*Error value*
Upper layer module request rejected	DDS_E_U_PDUID_REJECTED	0x10
Lower layer notify ignored	DDS_E_L_PDUID_IGNORED	0x11
CSM library error	DDS_E_CSM_LIBRARY_ERROR	0x30

▽

△

Type of error	*Related error code*	*Error value*
CSM check error	DDS_E_CSM_CHECK_FAILED	0x40
CRC check failed	DDS_E_CRC_CHECK_FAILED	0x41
Sample rejected	DDS_E_SAMPLE_REJECTED	0x42
Sample lost	DDS_E_SAMPLE_LOST	0x43
Timing constraints missed at receiver side	DDS_E_RECEIVER_TIMING_MISSED	0x44
Timing constraints missed at sender side	DDS_E_SENDER_TIMING_MISSED	0x45
Internal error	DDS_INTERNAL_ERROR	0x46

⌋

7.3.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_Dds_00801] Definition of imported datatypes of module Dds

Upstream requirements: FO_RS_Dds_00007

⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_VerifyResultType
SoAd	SoAd.h	SoAd_SoConIdType
StbM	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_TimeTupleType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_VirtualLocalTimeType
Std	Std_Types.h	Std_ReturnType
TcpIp	TcpIp.h	TcpIp_DomainType
	TcpIp.h	TcpIp_SockAddrType

⌋

8.2 Type definitions

8.2.1 Dds_ConfigType

[CP_SWS_Dds_00802] Definition of datatype Dds_ConfigType

Upstream requirements: FO_RS_Dds_00007, SRS_BSW_00405

⌈

Name	Dds_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	The content of the initialization data structure is implementation specific
Description	This is the type of the data structure containing the initialization data for Dds.	

▽

△

Available via	Dds.h

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Start-up and control services

8.3.1.1 Dds_Init

[CP_SWS_Dds_00810] Definition of API function Dds_Init

Upstream requirements: SRS_BSW_00405, SRS_BSW_00101, SRS_BSW_00414

⌈

Service Name	Dds_Init	
Syntax	`void Dds_Init ( const Dds_ConfigType* Dds_ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Dds_ConfigPtr	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes interfaces and variables of the AUTOSAR Dds layer.	
Available via	Dds.h	

⌋

[CP_SWS_Dds_00811] Dds_Init behaviour

Upstream requirements: SRS_BSW_00101

⌈The function `Dds_Init` shall initialize all module-related variables and constants according to the configuration.⌋

[CP_SWS_Dds_00812] Dds_Init - Entity state

Upstream requirements: SRS_BSW_00101

⌈The function `Dds_Init` shall initialize all Entities to "enabled" state.⌋

[CP_SWS_Dds_00813] Dds_Init - Queue state

Upstream requirements: SRS_BSW_00101

⌈The function `Dds_Init` shall empty all internal queues.⌋

8.3.1.2 Dds_ReleaseRxBuffer

[SWS_Dds_00844] Definition of API function Dds_ReleaseRxBuffer

Upstream requirements: FO_RS_Dds_00015

⌈

Service Name	Dds_ReleaseRxBuffer	

Syntax	`void Dds_ReleaseRxBuffer (`
	`  PduIdType RxPduId`
	`)`	

Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	RxPduId	Identifier of the received PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication from the upper layer to release the lower layer reception buffer.	
Available via	Dds.h	

⌋

8.3.2 Communication services

8.3.2.1 Dds_Transmit

[CP_SWS_Dds_00831] Definition of API function Dds_Transmit

Upstream requirements: SRS_BSW_00402, SRS_BSW_00411, SRS_BSW_00374, SRS_BSW_00379, SRS_BSW_00003, SRS_BSW_00318

⌈

Service Name	Dds_Transmit	

Syntax	`Std_ReturnType Dds_Transmit (`
	`  PduIdType TxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `E_NOT_OK`: Transmit request has not been accepted.
Description	Request for transmitting a message.	
Available via	Dds.h	

⌋

[CP_SWS_Dds_00852] Dds_Transmit - Error conditions

Upstream requirements: FO_RS_Dds_00015, SRS_BSW_00369

⌈The function `Dds_Transmit` shall call the `Det_ReportError`, if development error detection is enabled and if function call has failed because of the following reasons:

- `Dds` module is not initialized (`DDS_E_UNINIT`).
- `PduInfoPtr` equals `NULL_PTR` (`DDS_E_PARAM_POINTER`).
- Invalid upper layer `TxPduId` (`DDS_E_U_PDUID_INVALID`).

⌋

[CP_SWS_Dds_00854] Dds_Transmit - DDS_E_U_PDUID_REJECTED

Upstream requirements: FO_RS_Dds_00015

⌈If upper layer module transmission request cannot be accepted or the proper transmission queue is full, `Dds_Transmit` shall call the API `Det_ReportRuntimeError` with the runtime error code `DDS_E_U_PDUID_REJECTED` and then return `E_NOT_OK`.

⌋

[CP_SWS_Dds_00855] Dds_Transmit - E_OK

Upstream requirements: FO_RS_Dds_00015

⌈If upper layer module transmission request can be accepted, `Dds_Transmit` shall store the transmission upper layer PDU into the proper transmission queue, update the queue offset which indicates where to store new PDUs and return `E_OK`.⌋

8.3.2.2 Dds_RxIndication

[CP_SWS_Dds_00841] Definition of callback function Dds_RxIndication

Upstream requirements: FO_RS_Dds_00016

⌈

Service Name	Dds_RxIndication	
Syntax	`void Dds_RxIndication ( PduIdType RxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	Indication of a received PDU from the PDU Router interface module.
Available via	Dds.h

⌋

[CP_SWS_Dds_00861] Dds_RxIndication - Error conditions

Upstream requirements: FO_RS_Dds_00016

⌈The function `Dds_RxIndication` shall call the `Det_ReportError`, if development error detection is enabled and if function call has failed because of the following reasons:

- `Dds` module is not initialized (`DDS_E_UNINIT`)
- `PduInfoPtr` equals `NULL_PTR` (`DDS_E_PARAM_POINTER`).
- Invalid lower layer `RxPduId` (`DDS_E_L_PDUID_INVALID`).

⌋

[CP_SWS_Dds_00862] Dds_RxIndication - DDS_E_L_PDUID_IGNORED

Upstream requirements: FO_RS_Dds_00016

⌈If the reception lower layer PDU cannot be accepted or the proper internal queue is full, `Dds_RxIndication` shall call the API `Det_ReportRuntimeError` with the runtime error code `DDS_E_L_PDUID_IGNORED` and return.⌋

[CP_SWS_Dds_00863] Dds_RxIndication - OK condition

Upstream requirements: FO_RS_Dds_00016

⌈If the reception lower layer PDU can be accepted, the `Dds_RxIndication` shall store this PDU into the proper internal queue and return.⌋

8.3.2.3 Dds_TxConfirmation

[CP_SWS_Dds_00843] Definition of callback function Dds_TxConfirmation

Upstream requirements: FO_RS_Dds_00015

⌈

Service Name	Dds_TxConfirmation

Syntax	`void Dds_TxConfirmation (`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`

Service ID [hex]	0x40

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted successfully. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The PDU Router interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	Dds.h	

⌋

[CP_SWS_Dds_00871] Dds_TxConfirmation - Error conditions

Upstream requirements: FO_RS_Dds_00015

⌈The function Dds_TxConfirmation shall call the Det_ReportError, if development error detection is enabled and if function call has failed because of the following reasons:

- Dds module is not initialized (DDS_E_UNINIT)
- Invalid TxPduId (DDS_E_L_PDUID_INVALID).

⌋

[CP_SWS_Dds_00872] Dds_TxConfirmation behaviour

Upstream requirements: FO_RS_Dds_00015

⌈Dds_TxConfirmation shall call the API PduR_DdsTxConfirmation with the TxPduId of the upper layer PDU which caused the transmission and the same result parameter and return.⌋

8.3.2.4 Dds_TriggerTransmit

[CP_SWS_Dds_00835] Definition of callback function Dds_TriggerTransmit

Upstream requirements: FO_RS_Dds_00015

⌈

Service Name	Dds_TriggerTransmit

Syntax	`Std_ReturnType Dds_TriggerTransmit (`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`

Service ID [hex]	0x41
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Available via	Dds.h	

⌋

[CP_SWS_Dds_00881] Dds_TriggerTransmit - Error conditions

Upstream requirements: FO_RS_Dds_00015, SRS_BSW_00369

⌈The function Dds_TriggerTransmit shall call the Det_ReportError, if development error detection is enabled and if function call has failed because of the following reasons:

- Dds module is not initialized (DDS_E_UNINIT)
- PduInfoPtr equals NULL_PTR (DDS_E_PARAM_POINTER).
- Invalid lower layer TxPduId (DDS_E_L_PDUID_INVALID).

⌋

[CP_SWS_Dds_00882] Dds_TriggerTransmit behaviour

Upstream requirements: FO_RS_Dds_00015

⌈Within the function Dds_TriggerTransmit, the Dds shall copy the contents of its PDU transmit buffer to the PDU buffer given by PduInfoPtr->SduDataPtr and update PduInfoPtr->SduLength with length of the copied data accordingly.⌋

[CP_SWS_Dds_00883] Dds_TriggerTransmit - Error conditions

Upstream requirements: FO_RS_Dds_00015

⌈If another Dds_TriggerTransmit request on the same PduId is not yet completed, the function Dds_TriggerTransmit shall call the Det_ReportRuntimeError with the code DDS_E_L_PDUID_IGNORED.⌋

[CP_SWS_Dds_CONSTR_00884] Dds_TriggerTransmit limitation ⌈The Dds_TriggerTransmit function can be called only by lower-layer module. During validation of configuration, the Com configuration shall be checked. If any I-PDU belonging to Dds has a non NULL value for Com_TxTriggerTransmit, the validation shall return an error and the configuration shall be rejected as invalid.⌋

8.4 Scheduled functions

Following functions are called directly by Basic Software Scheduler. They have no return value and no parameter. All functions shall be non-reentrant

8.4.1 Dds_MainFunction_Rx

[CP_SWS_Dds_00823] Definition of scheduled function Dds_MainFunction_Rx

Upstream requirements: SRS_BSW_00424, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00478, SRS_BSW_00373

⌈

Service Name	Dds_MainFunction_Rx
Syntax	`void Dds_MainFunction_Rx ( void )`
Service ID [hex]	0x10
Description	Scheduled function of the Dds module for reception purpose
Available via	SchM_Dds.h

⌋

[CP_SWS_Dds_00826] Dds_MainFunction_Rx - Error conditions

Upstream requirements: FO_RS_Dds_00016

⌈If, during processing of the `Dds_MainFunction_Rx` any error or violation occurred, the `Dds_MainFunction_Rx` shall call the `Det_ReportRuntimeError` with the `DDS_INTERNAL_ERROR` code, drop received data and return.⌋

[CP_SWS_Dds_00827] Dds_MainFunction_Rx - OK conditions

Upstream requirements: FO_RS_Dds_00016

⌈If, during processing of the `Dds_MainFunction_Rx`, everything is ok, the `Dds_MainFunction_Rx` shall find the proper readers to manage the reception of data and shall call the API `PduR_DdsRxIndication` with the resulting upper layer PDU as input parameter.⌋

8.4.2 Dds_MainFunction_Tx

Into the `Dds_MainFunction_Tx` function all the DDS Middleware processing for transmission shall be performed. Internal DDS processing is out of the scope of the SWS: it is vendor-specific (as soon as the implementation is compliant with DDS OMG Specification ([2]). In the following section there are requirements needed to specify AUTOSAR APIs.

[CP_SWS_Dds_00824] Definition of scheduled function Dds_MainFunction_Tx

Upstream requirements: SRS_BSW_00424, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00478, SRS_BSW_00373

⌈

Service Name	Dds_MainFunction_Tx
Syntax	`void Dds_MainFunction_Tx ( void )`
Service ID [hex]	0x11
Description	Scheduled function of the Dds module for transmission purpose
Available via	SchM_Dds.h

⌋

[CP_SWS_Dds_00829] Dds_MainFunction_Tx - Error conditions

Upstream requirements: FO_RS_Dds_00015

⌈If, during processing of the `Dds_MainFunction_Tx` any error or violation occurred, the `Dds_MainFunction_Tx` shall call the `Det_ReportRuntimeError` with the `DDS_INTERNAL_ERROR` code, shall call the `PduR_DdsTxConfirmation` with result = `E_NOT_OK`, shall drop received data and return.⌋

[CP_SWS_Dds_00830] Dds_MainFunction_Tx - OK conditions

Upstream requirements: FO_RS_Dds_00015

⌈If, during processing of the `Dds_MainFunction_Tx`, everything is ok, the `Dds_MainFunction_Tx` shall find the proper writers to manage the transmission of data and shall call the API `PduR_DdsTransmit` with the resulting lower layer PDU as input parameter.⌋

8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.5.1 Mandatory interfaces

[CP_SWS_Dds_00832] Definition of mandatory interfaces required by module Dds

Upstream requirements: FO_RS_Dds_00005, FO_RS_Dds_00009, FO_RS_Dds_00010, FO_RS_Dds_00015, FO_RS_Dds_00016, SRS_BSW_00384, SRS_BSW_00385

⌈

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
PduR_DdsRxIndication	PduR_Dds.h	Indication of a received PDU from a lower layer communication interface module.
PduR_DdsTransmit	PduR_Dds.h	Requests transmission of a PDU.
PduR_DdsTxConfirmation	PduR_Dds.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.

⌋

8.5.2 Optional interfaces

[CP_SWS_Dds_00833] Definition of optional interfaces requested by module Dds

Upstream requirements: FO_RS_Dds_00005, FO_RS_Dds_00009, FO_RS_Dds_00010, SRS_BSW_00384, SRS_BSW_00385

⌈

API Function	Header File	Description
Crc_CalculateCRC32	Crc.h	This service makes a CRC32 calculation on Crc_Length data bytes.
Crc_CalculateCRC64	Crc.h	This service makes a CRC64 calculation on Crc_Length data bytes, using the polynomial 0x42F0E1EBA9EA3693. This CRC routine is used by E2E Profile 7.
Csm_MacGenerate	Csm.h	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.
Csm_MacVerify	Csm.h	Verifies the given MAC by comparing if the MAC is generated with the given data.
PduR_DdsReleaseRxBuffer (draft)	PduR_Dds.h	Indication from the upper layer to release the lower layer reception buffer.
SoAd_GetRemoteAddr	SoAd.h	Retrieves the remote address (IP address and port) actually used for the SoAd socket connection specified by SoConId
SoAd_OpenSoCon	SoAd.h	This service opens the socket connection specified by SoConId.

▽

△

API Function	*Header File*	*Description*
SoAd_ReleaseRemoteAddr	SoAd.h	By this API service the remote address (IP address and port) of the specified socket connection shall be released, i.e. set back to the configured remote address setting.
SoAd_SetRemoteAddr	SoAd.h	By this API service the remote address (IP address and port) of the specified socket connection shall be set.
SoAd_SetUniqueRemoteAddr	SoAd.h	This API service shall either return the socket connection index of the SoAdSocketConnection Group where the specified remote address (IP address and port) is set or assign the remote address to an unused socket connection from the same SoAdSocketConnectionGroup.
StbM_GetCurrentTime	StbM.h	Returns a time tuple (Local time, Global time and Timebase status) and user data details Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).

⌋

8.5.3 Configurable interfaces

None.

9 Sequence diagrams

9.1 Transmission

9.1.1 Dds message transmission

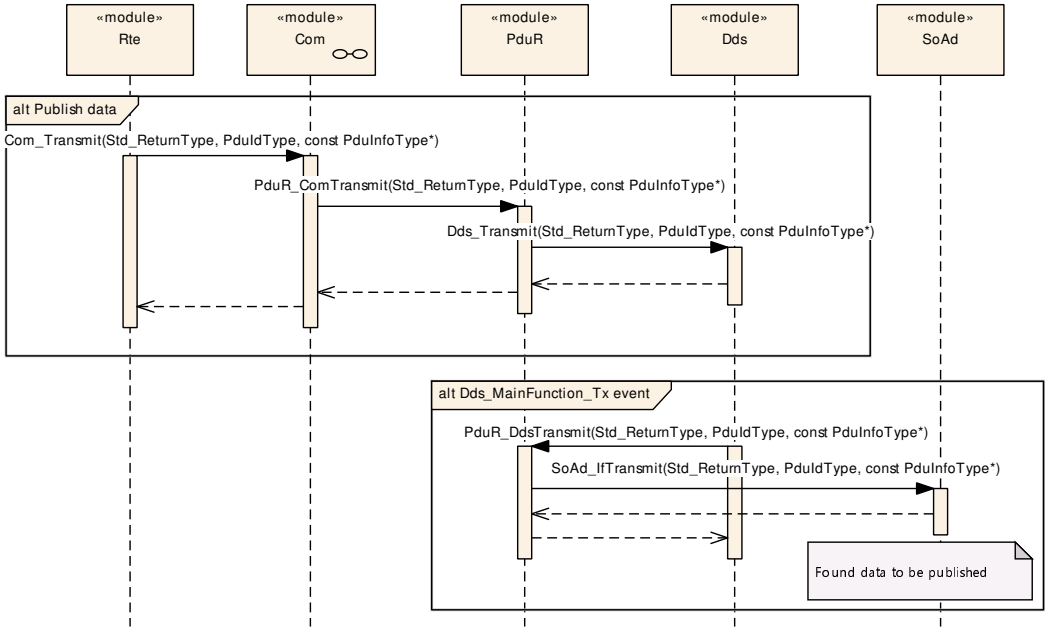


Figure 9.1: Dds transmission path

9.1.2 Dds message transmission confirmation

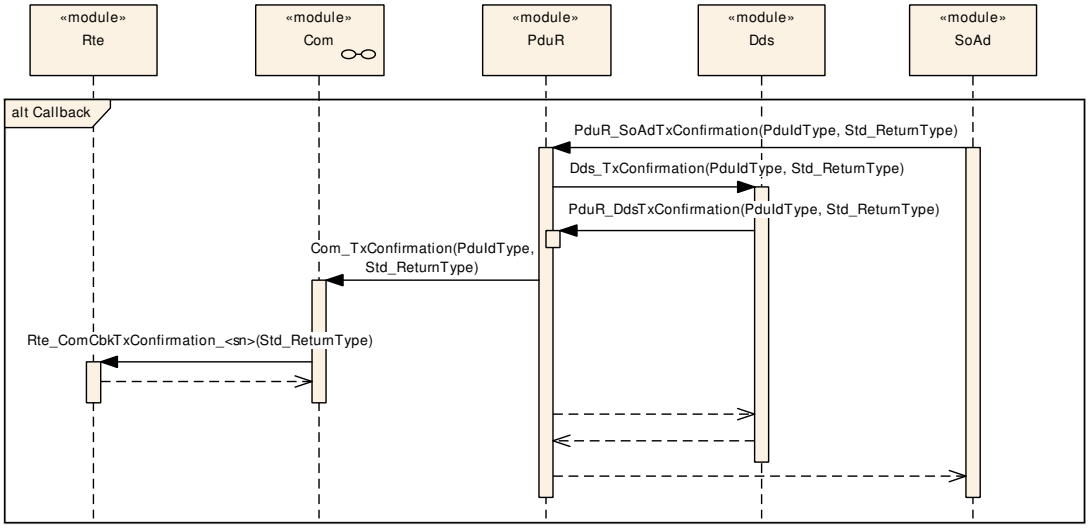


Figure 9.2: Dds transmission confirmation path

9.1.3 Dds message trigger transmission

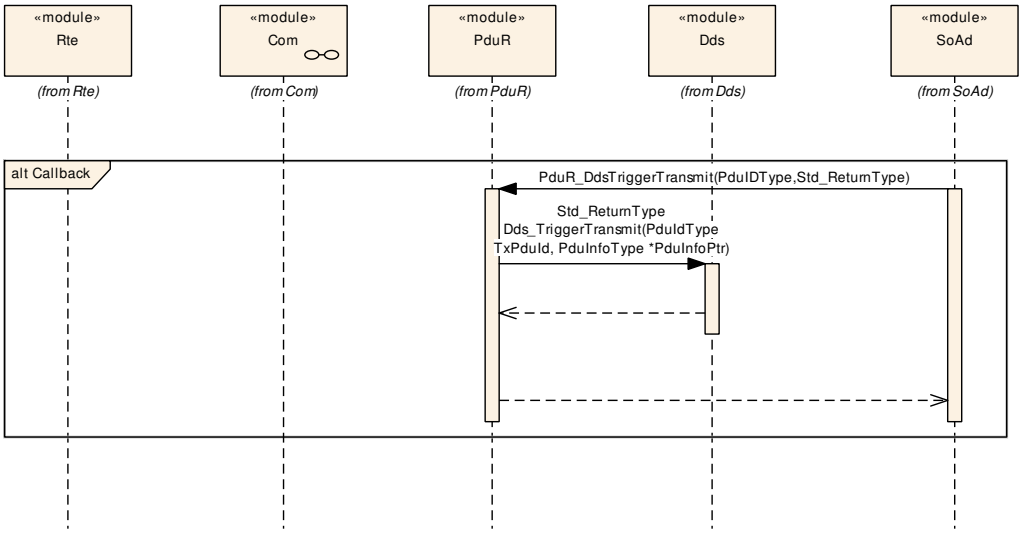


Figure 9.3: Dds trigger transmission path

9.2 Reception

9.2.1 Dds received indication event

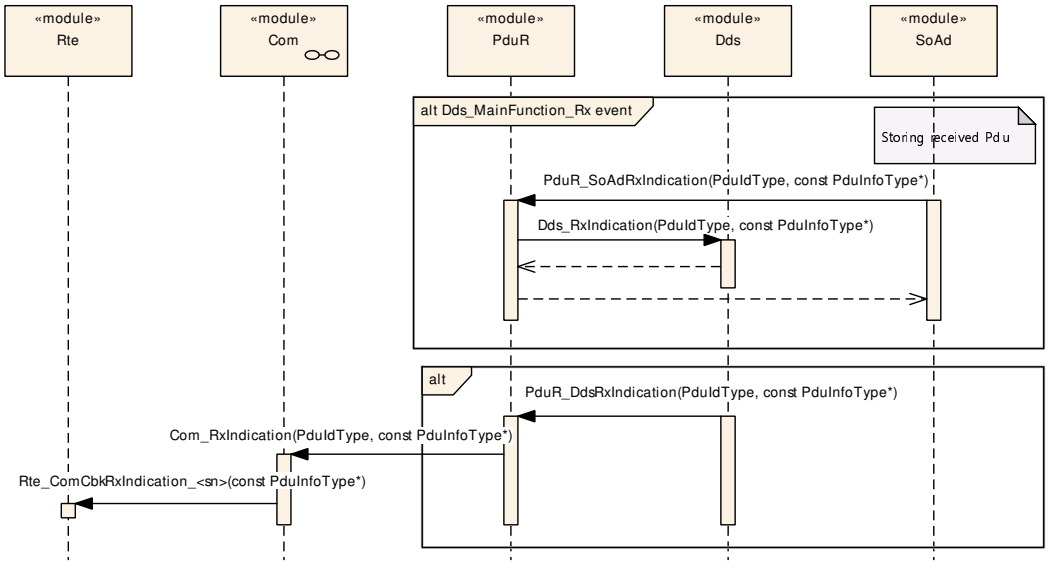


Figure 9.4: Dds reception path

9.3 Discovery

9.3.1 Remote endpoint address setting

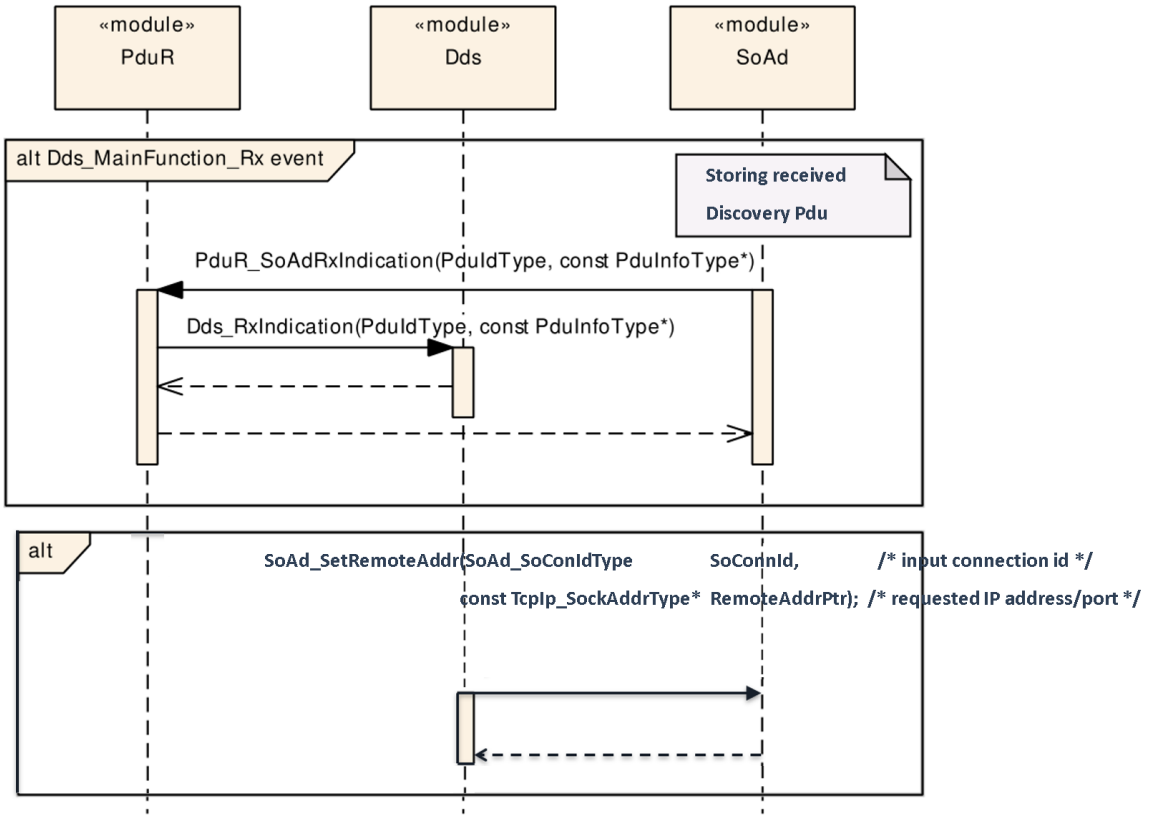


Figure 9.5: Remote Participant discovery event and following setting of the IP Address/-port

9.3.2 Setting SoAdConnection to unused

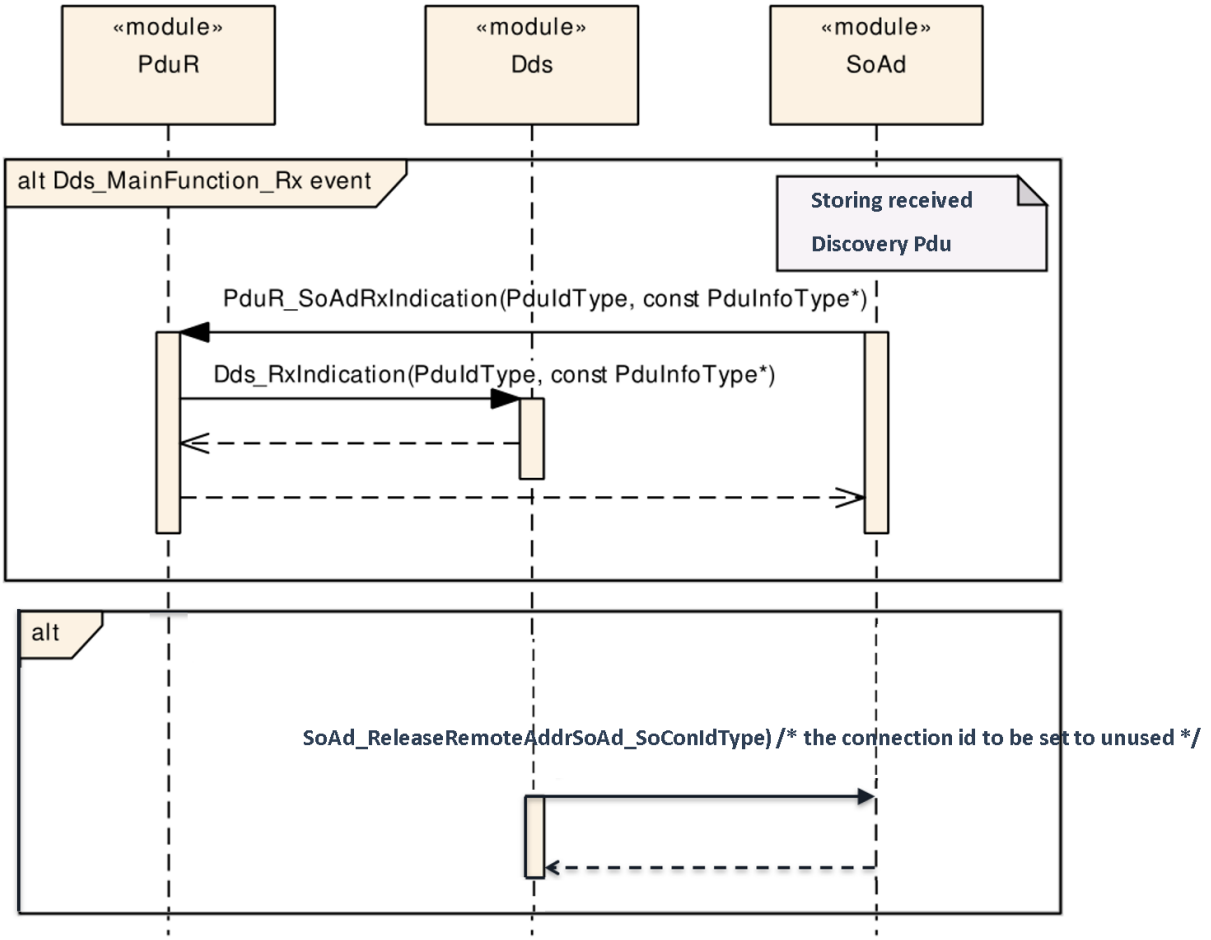


Figure 9.6: Remote Participant discovery event removal and following setting of the related connection to unused

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module Dds.

Chapter 10.3 specifies published information of the module Dds.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters.

10.2.1 Dds

[ECUC_Dds_00001] Definition of EcucModuleDef Dds ⌈

Module Name	Dds
Description	Configuration of the Dds module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR Dds module.
DdsGeneral	1	This container lists the general configuration parameters for the Dds module.

⌋

In the picture below, the UML diagram of `Dds` is shown:

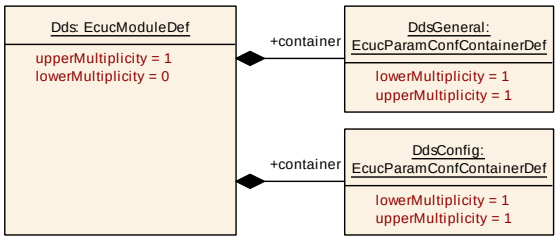


Figure 10.1: Dds model

10.2.2 Dds General

[ECUC_Dds_00002] Definition of EcucParamConfContainerDef DdsGeneral ⌈

Container Name	DdsGeneral
Parent Container	Dds
Upstream mapping defined	–
Description	This container lists the general configuration parameters for the Dds module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDevErrorDetect	1	[ECUC_Dds_00003]
DdsMainRxFunctionPeriod	1	[ECUC_Dds_00004]
DdsMainTxFunctionPeriod	1	[ECUC_Dds_00127]
DdsSynchronizedTimeBaseRef	0..1	[ECUC_Dds_00128]

No Included Containers

⌋

[ECUC_Dds_00003] Definition of EcucBooleanParamDef DdsDevErrorDetect ⌈

Parameter Name	DdsDevErrorDetect
Parent Container	DdsGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Dds_00004] Definition of EcucFloatParamDef DdsMainRxFunctionPeriod ⌈

Parameter Name	DdsMainRxFunctionPeriod	
Parent Container	DdsGeneral	
Upstream mapping defined	–	
Description	This parameter defines the cycle time in seconds of the periodic call of the Dds_Main Function_Rx.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00127] Definition of EcucFloatParamDef DdsMainTxFunctionPeriod ⌈

Parameter Name	DdsMainTxFunctionPeriod	
Parent Container	DdsGeneral	
Upstream mapping defined	–	
Description	This parameter defines the cycle time in seconds of the periodic call of the Dds_Main Function_Tx.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00128] Definition of EcucReferenceDef DdsSynchronizedTimeBaseRef ⌈

Parameter Name	DdsSynchronizedTimeBaseRef
Parent Container	DdsGeneral
Upstream mapping defined	–
Description	Reference to a StbM Synchronized Time Base.
Multiplicity	0..1
Type	Symbolic name reference to StbMSynchronizedTimeBase
Pre-Compile only	true

⌋

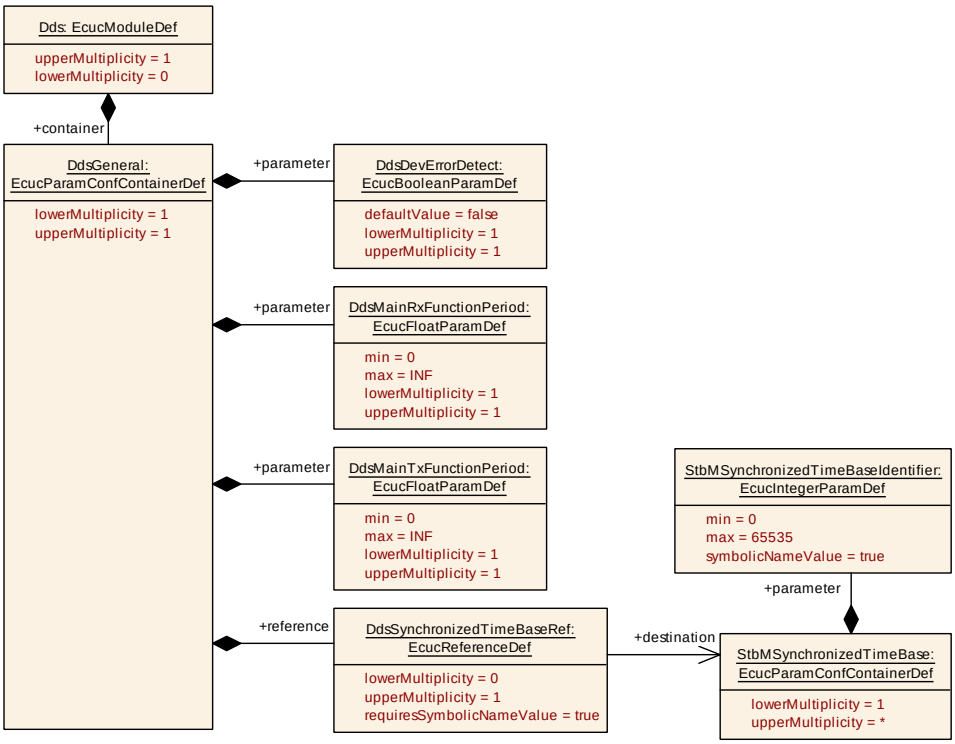


Figure 10.2: Dds General

10.2.3 Dds Config

[ECUC_Dds_00005] Definition of EcucParamConfContainerDef DdsConfig ⌈

Container Name	DdsConfig
Parent Container	Dds
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Dds module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsAppDataRxPduCollection	0..1	Collection of upper layer Rx PDUs towards the application layer.
DdsAppDataTxPduCollection	0..1	Collection of upper layer Tx PDUs towards the application layer.
DdsDomainDpMulticastRtpsPdu Collection	0..1	Collection of low-level multicast PDUs used for transporting SPDP discovery traffic.
DdsDomainMulticastRtpsPdu Collection	0..1	Collection of low-level multicast PDUs used for transporting user traffic.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDomainParticipantCollection	0..1	Collection of DDS Domain Participants.
DdsRxQueueCollection	0..1	Collection of Rx queues.
DdsTxQueueCollection	0..1	Collection of Tx queues.

⌋

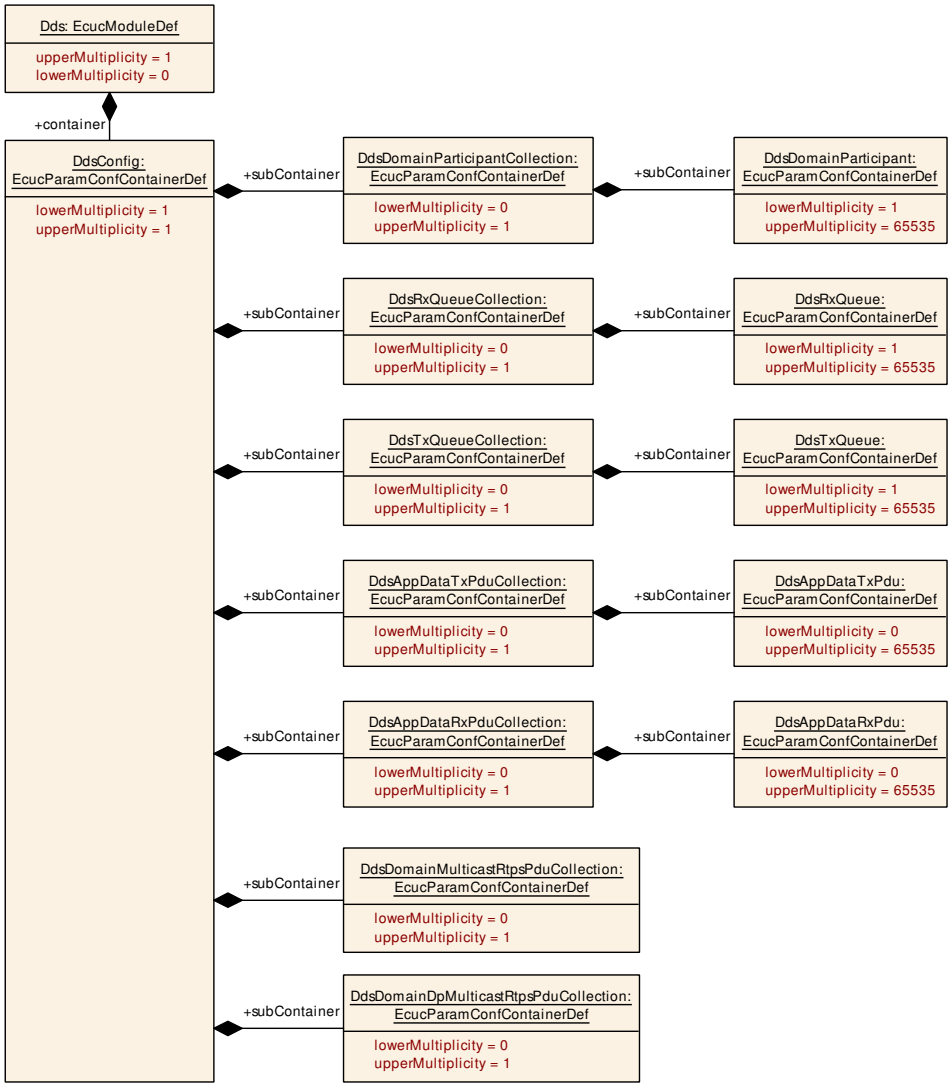


Figure 10.3: Dds Config

10.2.3.1 DdsAppDataPduCollection

The `DdsAppDataTxPduCollection` and `DdsAppDataRxPduCollection` containers model the pool of all the upper layer PDUs (respectively Tx and Rx) used for interaction between application layers and the Dds module. They are used just to have

a unique definition points for all the upper layer PDUs (they are simply containers of containers).

In the picture below, the UML diagram of DdsAppDataTxPduCollection and DdsAppDataRxPduCollection is shown:

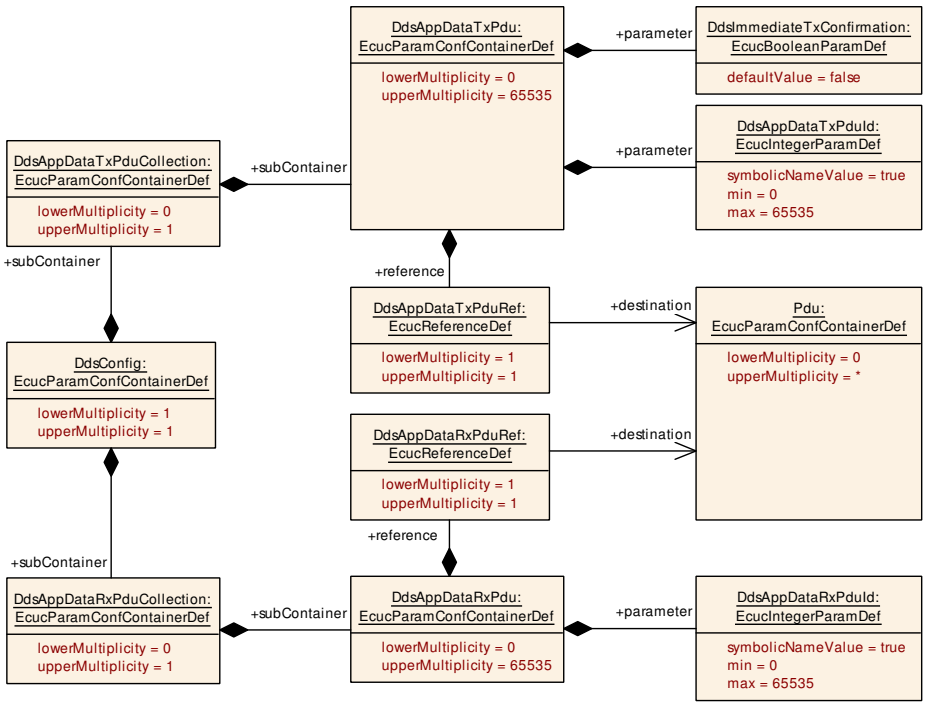


Figure 10.4: Dds Upper layer Pdus

10.2.3.1.1 DdsAppDataTxPduCollection

[ECUC_Dds_00131] Definition of EcucParamConfContainerDef DdsAppDataTxPduCollection ⌈

Container Name	DdsAppDataTxPduCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of upper layer Tx PDUs towards the application layer.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsAppDataTxPdu	0..65535	The upper layer PDU used to transmit data from Application to DDS itself.

⌋

10.2.3.1.1.1 DdsAppDataTxPdu

[ECUC_Dds_00132] Definition of EcucParamConfContainerDef DdsAppDataTx Pdu ⌈

Container Name	DdsAppDataTxPdu	
Parent Container	DdsAppDataTxPduCollection	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The upper layer PDU used to transmit data from Application to DDS itself.	
Multiplicity	0..65535	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsAppDataTxPduId	1	[ECUC_Dds_00133]
DdsImmediateTxConfirmation	1	[ECUC_Dds_00230]
DdsAppDataTxPduRef	1	[ECUC_Dds_00134]

No Included Containers

⌋

[ECUC_Dds_00133] Definition of EcucIntegerParamDef DdsAppDataTxPduId ⌈

Parameter Name	DdsAppDataTxPduId	
Parent Container	DdsAppDataTxPdu	
Upstream mapping defined	–	
Description	The current pdu local id.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00230] Definition of EcucBooleanParamDef DdsImmediateTxConfirmation ⌈

Parameter Name	DdsImmediateTxConfirmation
Parent Container	DdsAppDataTxPdu
Upstream mapping defined	–
Description	Switches the behavior of the TxConfirmation to the upper layer: • True: Immediately after the provided data was copied to the internal buffer. • False: After a TxConfirmation from the lower layer was received.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	false
Pre-Compile only	true

⌋

[ECUC_Dds_00134] Definition of EcucReferenceDef DdsAppDataTxPduRef ⌈

Parameter Name	DdsAppDataTxPduRef
Parent Container	DdsAppDataTxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the Dds module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.2.3.1.2 DdsAppDataRxPduCollection

[ECUC_Dds_00178] Definition of EcucParamConfContainerDef DdsAppDataRxPduCollection ⌈

Container Name	DdsAppDataRxPduCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of upper layer Rx PDUs towards the application layer.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsAppDataRxPdu	0..65535	The upper layer PDU used to send received data from DDS to Application.

⌋

10.2.3.1.2.1 DdsAppDataRxPdu

[ECUC_Dds_00135] Definition of EcucParamConfContainerDef DdsAppDataRxPdu ⌈

Container Name	DdsAppDataRxPdu	
Parent Container	DdsAppDataRxPduCollection	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The upper layer PDU used to send received data from DDS to Application.	
Multiplicity	0..65535	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsAppDataRxPduId	1	[ECUC_Dds_00136]
DdsAppDataRxPduRef	1	[ECUC_Dds_00137]

No Included Containers

⌋

[ECUC_Dds_00136] Definition of EcucIntegerParamDef DdsAppDataRxPduId ⌈

Parameter Name	DdsAppDataRxPduId	
Parent Container	DdsAppDataRxPdu	
Upstream mapping defined	–	
Description	The current pdu local id.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00137] Definition of EcucReferenceDef DdsAppDataRxPduRef ⌈

Parameter Name	DdsAppDataRxPduRef
Parent Container	DdsAppDataRxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the Dds module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.2.3.2 DdsQueueCollection

The `DdsTxQueueCollection` and `DdsRxQueueCollection` containers model the pool of all the queue used respectively to store:

- DdsAppDataTxPdus from PduR (as upper) to DDS
- DdsRtpsDataRxPdus or DdsRtpsMulticastDataRxPdus from PduR (as lower) to DDS.

There are used just to have a unique definition points for all the queues (they are simply containers of containers).

10.2.3.2.1 DdsRxQueueCollection

The DdsRxQueueCollection container is used to collect DdsRxQueues.

[ECUC_Dds_00180] Definition of EcucParamConfContainerDef DdsRxQueueCollection ⌈

Container Name	DdsRxQueueCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of Rx queues.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsRxQueue	1..65535	The queue used to save DdsRtpsDataRxPdus from PduR (as lower) to DDS. One single queue can be used to save one or more DdsRtps DataRxPdus.

⌋

10.2.3.2.1.1 DdsRxQueue

The DdsRxQueue containiner models the queues used to store DdsRtpsDataRxPdus or DdsRtpsMulticastDataRxPdus from PduR (as lower) to DDS.

Note: One single queue can be used to save one or more DdsRtpsDataRxPdus or DdsRtpsMulticastDataRxPdu.

The processing of those queues is up to DDS middleware, according its own internal policies (QoS policies, DdsDataReaders subscribed etc.).

In the picture below, the UML diagram of DdsRxQueue template is shown:

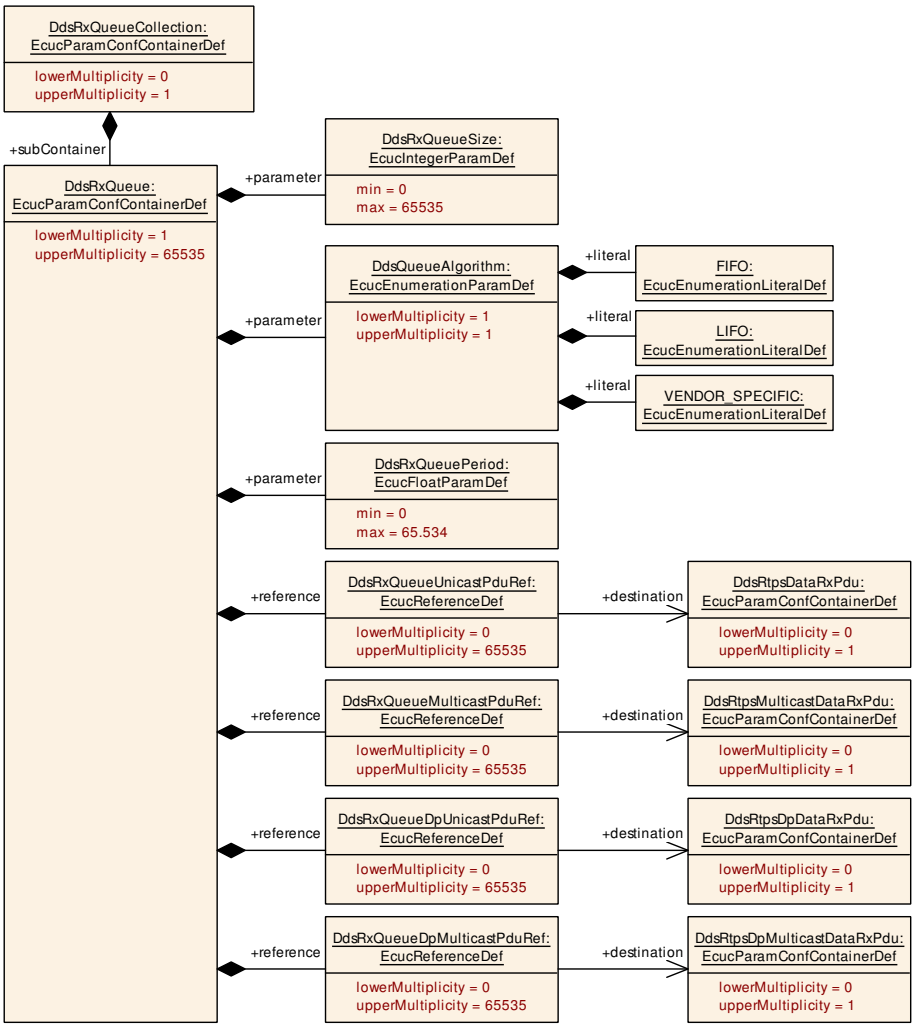


Figure 10.5: DdsRxQueue

[ECUC_Dds_00167] Definition of EcucParamConfContainerDef DdsRxQueue ⌈

Container Name	DdsRxQueue
Parent Container	DdsRxQueueCollection
Upstream mapping defined	–
Description	The queue used to save DdsRtpsDataRxPdus from PduR (as lower) to DDS. One single queue can be used to save one or more DdsRtpsDataRxPdus.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsQueueAlgorithm	1	[ECUC_Dds_00170]
DdsRxQueuePeriod	1	[ECUC_Dds_00169]
DdsRxQueueSize	1	[ECUC_Dds_00168]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRxQueueDpMulticastPduRef	0..65535	[ECUC_Dds_00192]
DdsRxQueueDpUnicastPduRef	0..65535	[ECUC_Dds_00191]
DdsRxQueueMulticastPduRef	0..65535	[ECUC_Dds_00172]
DdsRxQueueUnicastPduRef	0..65535	[ECUC_Dds_00171]

No Included Containers

⌋

[ECUC_Dds_00170] Definition of EcucEnumerationParamDef DdsQueueAlgorithm ⌈

Parameter Name	DdsQueueAlgorithm	
Parent Container	DdsRxQueue, DdsTxQueue	
Upstream mapping defined	–	
Description	Single queue scheduling algorithm	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FIFO	FIFO
	LIFO	LIFO
	VENDOR_SPECIFIC	VENDOR_SPECIFIC
Pre-Compile only	true	

⌋

[ECUC_Dds_00169] Definition of EcucFloatParamDef DdsRxQueuePeriod ⌈

Parameter Name	DdsRxQueuePeriod	
Parent Container	DdsRxQueue	
Upstream mapping defined	–	
Description	Scheduling period of the single queue. Time given in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00168] Definition of EcucIntegerParamDef DdsRxQueueSize ⌈

Parameter Name	DdsRxQueueSize
Parent Container	DdsRxQueue
Upstream mapping defined	–
Description	Queue size in bytes

▽

△

Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00192] Definition of EcucReferenceDef DdsRxQueueDpMulticastPduRef ⌈

Parameter Name	DdsRxQueueDpMulticastPduRef
Parent Container	DdsRxQueue
Upstream mapping defined	–
Description	Reference to a reception service discovered multicast lower layer PDU to be stored in the given queue.
Multiplicity	0..65535
Type	Reference to DdsRtpsDpMulticastDataRxPdu
Pre-Compile only	true

⌋

[ECUC_Dds_00191] Definition of EcucReferenceDef DdsRxQueueDpUnicastPduRef ⌈

Parameter Name	DdsRxQueueDpUnicastPduRef
Parent Container	DdsRxQueue
Upstream mapping defined	–
Description	Reference to a reception unicast service discovered lower layer Pdu to be stored in the given queue.
Multiplicity	0..65535
Type	Reference to DdsRtpsDpDataRxPdu
Pre-Compile only	true

⌋

[ECUC_Dds_00172] Definition of EcucReferenceDef DdsRxQueueMulticastPduRef ⌈

Parameter Name	DdsRxQueueMulticastPduRef
Parent Container	DdsRxQueue
Upstream mapping defined	–
Description	Reference to a reception multicast lower layer PDU to be stored in the given queue.
Multiplicity	0..65535
Type	Reference to DdsRtpsMulticastDataRxPdu
Pre-Compile only	true

⌋

[ECUC_Dds_00171] Definition of EcucReferenceDef DdsRxQueueUnicastPduRef
⌈

Parameter Name	DdsRxQueueUnicastPduRef
Parent Container	DdsRxQueue
Upstream mapping defined	–
Description	Reference to a reception unicast lower layer Pdu to be stored in the given queue.
Multiplicity	0..65535
Type	Reference to DdsRtpsDataRxPdu
Pre-Compile only	true

⌋

10.2.3.2.2 DdsTxQueueCollection

The DdsTxQueueCollection container is used to collect DdsTxQueues.

[ECUC_Dds_00181] Definition of EcucParamConfContainerDef DdsTxQueueCollection ⌈

Container Name	DdsTxQueueCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of Tx queues.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsTxQueue	1..65535	The queue used to save DdsAppDataTxPdus from PduR (as upper) to DDS. One single queue can be used to save one or more DdsAppDataTxPdus.

⌋

10.2.3.2.2.1 DdsTxQueue

The DdsTxQueue containiner models the queues used to save DdsAppDataTxPdus from PduR (as upper) to DDS.

Note: One single DdsTxQueue can be used to save one or more DdsAppDataTxPdus.

The processing of those queues is up to DDS middleware, according its own internal policies (QoS policies, DdsDataWriters to be published etc.).

In the picture below, the UML diagram of DdsTxQueue template is shown:

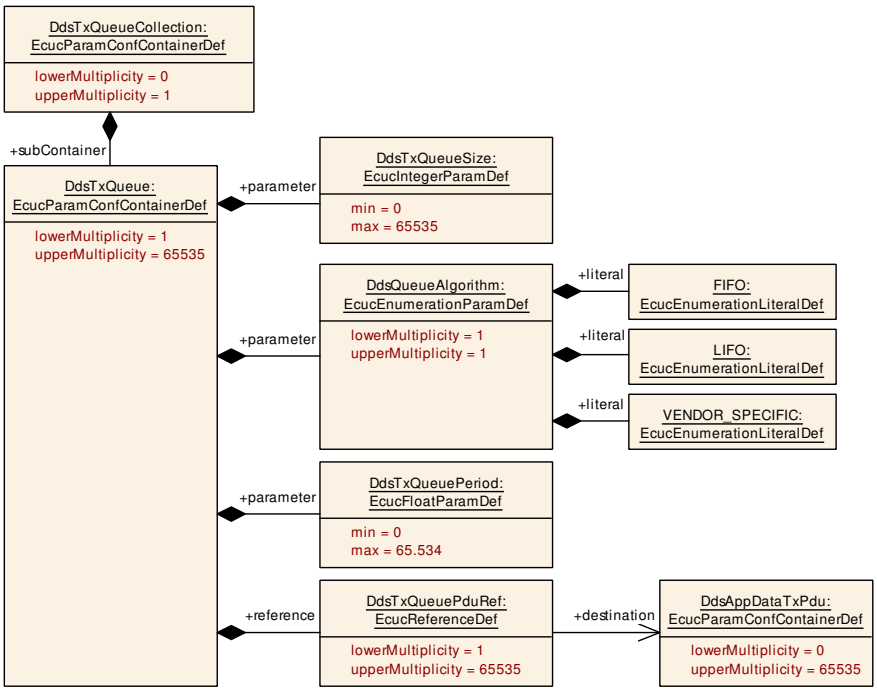


Figure 10.6: DdsTxQueue

[ECUC_Dds_00173] Definition of EcucParamConfContainerDef DdsTxQueue ⌈

Container Name	DdsTxQueue
Parent Container	DdsTxQueueCollection
Upstream mapping defined	–
Description	The queue used to save DdsAppDataTxPdus from PduR (as upper) to DDS. One single queue can be used to save one or more DdsAppDataTxPdus.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsQueueAlgorithm	1	[ECUC_Dds_00170]
DdsTxQueuePeriod	1	[ECUC_Dds_00175]
DdsTxQueueSize	1	[ECUC_Dds_00174]
DdsTxQueuePduRef	1..65535	[ECUC_Dds_00176]

No Included Containers

⌋

For parameter table [ECUC_Dds_00170] DdsQueueAlgorithm, see definition below container DdsRxQueue.

[ECUC_Dds_00175] Definition of EcucFloatParamDef DdsTxQueuePeriod ⌈

Parameter Name	DdsTxQueuePeriod
Parent Container	DdsTxQueue
Upstream mapping defined	–
Description	Scheduling period of the single queue. Time given in seconds.
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. 65.534]
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00174] Definition of EcucIntegerParamDef DdsTxQueueSize ⌈

Parameter Name	DdsTxQueueSize
Parent Container	DdsTxQueue
Upstream mapping defined	–
Description	Queue size in bytes
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65535
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00176] Definition of EcucReferenceDef DdsTxQueuePduRef ⌈

Parameter Name	DdsTxQueuePduRef
Parent Container	DdsTxQueue
Upstream mapping defined	–
Description	Reference to a transmission upper layer PDU to be stored in the given queue.
Multiplicity	1..65535
Type	Reference to DdsAppDataTxPdu
Pre-Compile only	true

⌋

10.2.3.3 DdsDomainParticipantCollection

The `DdsDomainParticipantCollection` container models the pool of all the `DdsDomainParticipant`. It is used just to have a unique definition points for all the `DdsDomainParticipant`s (it is simply a container of containers).

In the picture below, the UML diagram of `DdsDomainParticipantCollection` container is shown.

[ECUC_Dds_00179] Definition of EcucParamConfContainerDef DdsDomainParticipantCollection ⌈

Container Name	DdsDomainParticipantCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of DDS Domain Participants.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDomainParticipant	1..65535	This container represents the configuration of one single Domain Participant hosted within the current node. One node can contain more than one Domain Participant.

⌋

10.2.3.3.1 DdsDomainParticipant

In the picture below, the UML diagram of `DdsDomainParticipant` container is shown.

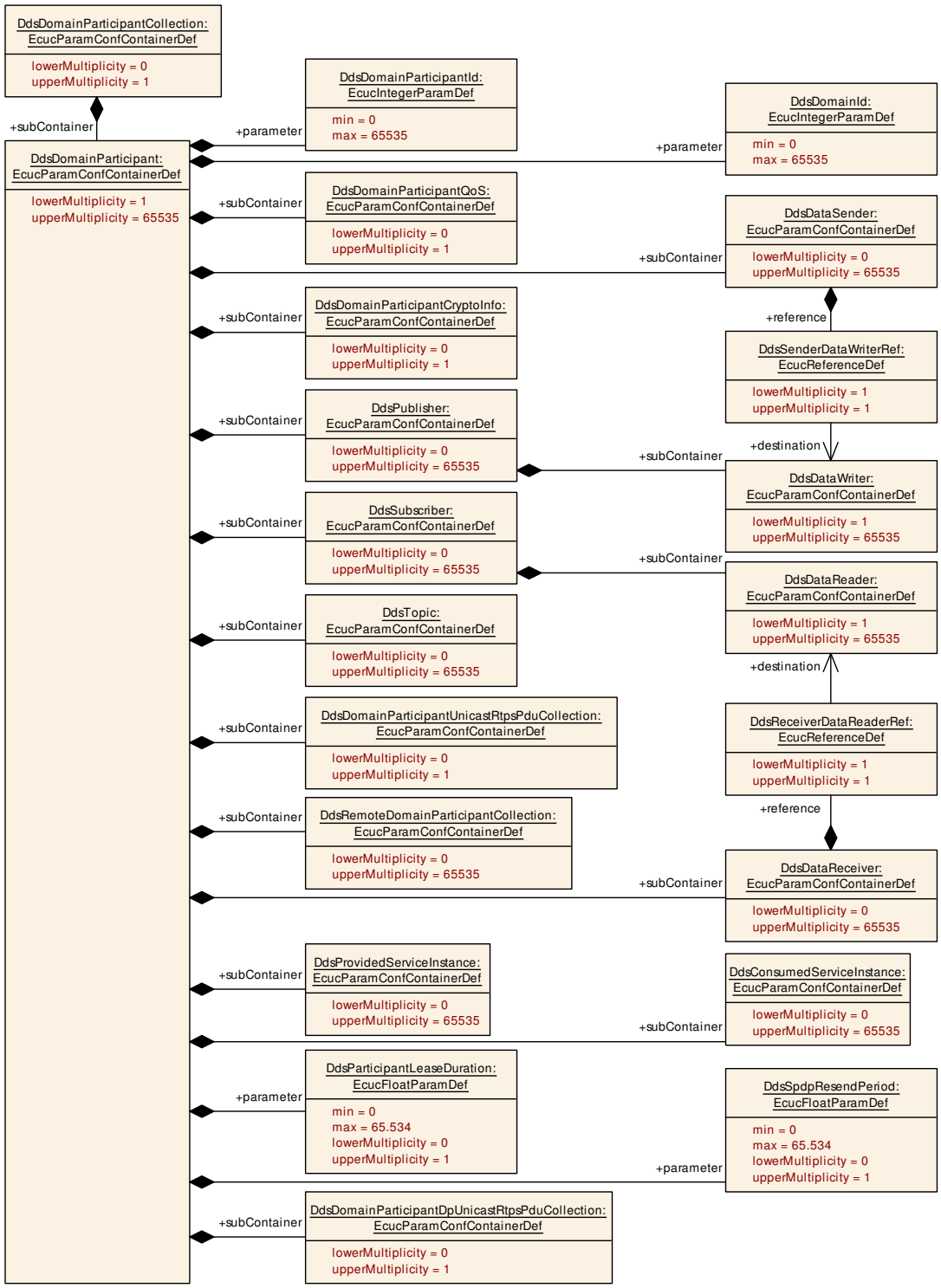


Figure 10.7: Dds Domain Participant

[ECUC_Dds_00012] Definition of EcucParamConfContainerDef DdsDomainParticipant ⌈

Container Name	DdsDomainParticipant
Parent Container	DdsDomainParticipantCollection
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of one single Domain Participant hosted within the current node. One node can contain more than one Domain Participant.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDomainId	1	[ECUC_Dds_00138]
DdsDomainParticipantId	1	[ECUC_Dds_00014]
DdsParticipantLeaseDuration	0..1	[ECUC_Dds_00228]
DdsSpdpResendPeriod	0..1	[ECUC_Dds_00229]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsConsumedServiceInstance	0..65535	This container represents the configuration of consumed service instance at Ecuc level in a concrete implementation on top of DDS.
DdsDataReceiver	0..65535	Container useful to usability: it defines a DDS DataReader linked to a Sender/ReceiverInterface.
DdsDataSender	0..65535	Container useful to usability: it defines a DDS DataWriter linked to a Sender/ReceiverInterface.
DdsDomainParticipantCryptoInfo	0..1	This container contains the configuration of the Crypto service to be used by Entities belonging to this DomainParticipant. If it is not present, it means that not security mechanism is supported.
DdsDomainParticipantDpUnicastRtpsPduCollection	0..1	Collection of unicast lower layer PDUs used in service discoved communication towards lower network layers.
DdsDomainParticipantQoS	0..1	This container represents the configuration of QoS supported by the Dds DomainParticipant.
DdsDomainParticipantUnicastRtpsPduCollection	0..1	Collection of low-level unicast PDUs used for transporting both user traffic and discovery traffic.
DdsProvidedServiceInstance	0..65535	This container represents the configuration of a provided service instance at Ecuc level in a concrete implementation on top of DDS.
DdsPublisher	0..65535	This container represents the configuration of one Publisher.
DdsRemoteDomainParticipantCollection	0..65535	Collection of Remote Domain Participants.
DdsSubscriber	0..65535	This container represents the configuration of a Subscriber.
DdsTopic	0..65535	This container represents the configuration of one Topic.

⌋

[ECUC_Dds_00138] Definition of EcucIntegerParamDef DdsDomainId ⌈

Parameter Name	DdsDomainId
Parent Container	DdsDomainParticipant, DdsRtpsDpMulticastDataRxPdu, DdsRtpsDpMulticastDataTxPdu, DdsRtpsMulticastDataRxPdu, DdsRtpsMulticastDataTxPdu

▽

△

Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The ID of the Domain to which this DDS node belongs. It unambiguously identifies the DDS Domain to which the DomainParticipant belongs. Note: Only entities that belong to the same DDS Domain can communicate with each other.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00014] Definition of EcucIntegerParamDef DdsDomainParticipantId

⌈

Parameter Name	DdsDomainParticipantId	
Parent Container	DdsDomainParticipant	
Upstream mapping defined	–	
Description	Identifier of a Dds Domain Participant. Dependency: inter-ECU - this value shall be shared between configurations.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00228] Definition of EcucFloatParamDef DdsParticipantLeaseDuration ⌈

Parameter Name	DdsParticipantLeaseDuration	
Parent Container	DdsDomainParticipant	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The periodic rate to send SPDP announcement message. Time given in seconds.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00229] Definition of EcucFloatParamDef DdsSpdpResendPeriod ⌈

Parameter Name	DdsSpdpResendPeriod
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	How long a Participant should be considered alive every time an announcement is received from the Participant. If a Participant fails to send another announcement within this time period, the Participant can be considered gone. Time given in seconds.
Multiplicity	0..1
Type	EcucFloatParamDef
Range	[0 .. 65.534]
Default value	–
Pre-Compile only	true

⌋

10.2.3.3.1.1 DdsDomainParticipantUnicastRtpsPduCollection

The DdsDomainParticipantUnicastRtpsPduCollection container models the pool of all the unicast lower layer PDUs (both Tx and Rx) towards lower network layers.

It is used just to have a unique definition points for all the unicast lower layer PDUs (it is simply a container of containers).

In the picture below, the UML diagram of DdsDomainParticipantUnicastRtpsPduCollection container is shown.

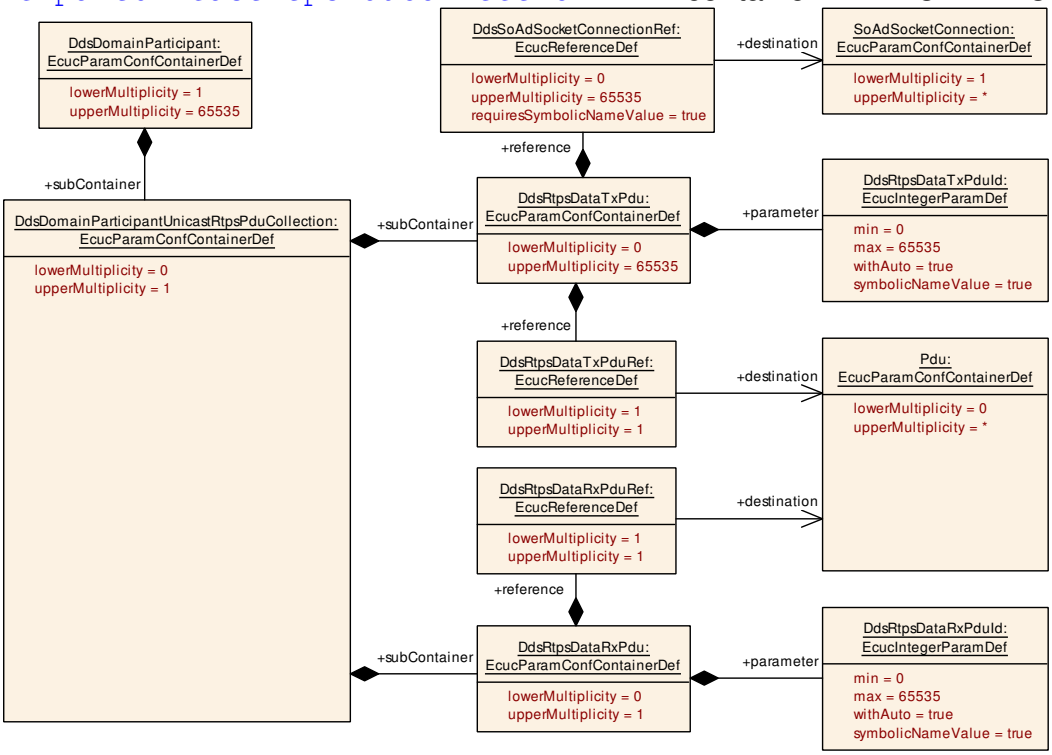


Figure 10.8: DdsDomainParticipantUnicastRtpsPduCollection

[ECUC_Dds_00143] Definition of EcucParamConfContainerDef DdsDomainParticipantUnicastRtpsPduCollection ⌈

Container Name	DdsDomainParticipantUnicastRtpsPduCollection
Parent Container	DdsDomainParticipant
Upstream mapping defined	–
Description	Collection of low-level unicast PDUs used for transporting both user traffic and discovery traffic.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsRtpsDataRxPdu	0..1	The unicast reception lower layer pdu used to send data from the lower network layer to DDS itself.
DdsRtpsDataTxPdu	0..65535	The unicast transmission lower layer pdu used to transmit data from DDS to lower network layer.

⌋

DdsRtpsDataRxPdu

[ECUC_Dds_00148] Definition of EcucParamConfContainerDef DdsRtpsDataRxPdu ⌈

Container Name	DdsRtpsDataRxPdu	
Parent Container	DdsDomainParticipantUnicastRtpsPduCollection	
Upstream mapping defined	–	
Description	The unicast reception lower layer pdu used to send data from the lower network layer to DDS itself.	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRtpsDataRxPduId	1	[ECUC_Dds_00149]
DdsRtpsDataRxPduRef	1	[ECUC_Dds_00150]

No Included Containers

⌋

[ECUC_Dds_00149] Definition of EcucIntegerParamDef DdsRtpsDataRxPduId ⌈

Parameter Name	DdsRtpsDataRxPduId	
Parent Container	DdsRtpsDataRxPdu, DdsRtpsDpDataRxPdu	
Upstream mapping defined	–	
Description	The current pdu local id.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Dds_00150] Definition of EcucReferenceDef DdsRtpsDataRxPduRef ⌈

Parameter Name	DdsRtpsDataRxPduRef
Parent Container	DdsRtpsDataRxPdu, DdsRtpsDpDataRxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the Dds module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

DdsRtpsDataTxPdu

[ECUC_Dds_00145] Definition of EcucParamConfContainerDef DdsRtpsDataTx Pdu ⌈

Container Name	DdsRtpsDataTxPdu	
Parent Container	DdsDomainParticipantUnicastRtpsPduCollection	
Upstream mapping defined	–	
Description	The unicast transmission lower layer pdu used to transmit data from DDS to lower network layer.	
Multiplicity	0..65535	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRtpsDataTxPduId	1	[ECUC_Dds_00146]
DdsRtpsDataTxPduRef	1	[ECUC_Dds_00147]
DdsSoAdSocketConnectionRef	0..65535	[ECUC_Dds_00185]

No Included Containers

⌋

[ECUC_Dds_00146] Definition of EcucIntegerParamDef DdsRtpsDataTxPduId ⌈

Parameter Name	DdsRtpsDataTxPduId	
Parent Container	DdsRtpsDataTxPdu, DdsRtpsDpDataTxPdu	
Upstream mapping defined	–	
Description	The current pdu local id	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Dds_00147] Definition of EcucReferenceDef DdsRtpsDataTxPduRef ⌈

Parameter Name	DdsRtpsDataTxPduRef
Parent Container	DdsRtpsDataTxPdu, DdsRtpsDpDataTxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the Dds module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Dds_00185] Definition of EcucReferenceDef DdsSoAdSocketConnection Ref ⌈

Parameter Name	DdsSoAdSocketConnectionRef
Parent Container	DdsRtpsDataTxPdu, DdsRtpsDpDataTxPdu, DdsRtpsDpMulticastDataTxPdu, DdsRtps MulticastDataTxPdu
Upstream mapping defined	–
Description	NOTE: this is required in case to use SPDP/SEDP because the Dds BSW needs to know which is the relationship between the DdsRtpsDataTxPdu and the SoAd Connection (e.g the remoteParticipantIpAddress, remoteParticipantPort) the PDU wants to use. In the current model the problem could be bypassed using only one Dds RtpsDataTxPdu and PDU meta-data, but using only one PDU the Dds is not able to parallelize the trasmission (See APPENDIX F.1)
Multiplicity	0..65535
Type	Symbolic name reference to SoAdSocketConnection
Pre-Compile only	true

⌋

10.2.3.3.1.2 DdsDomainParticipantDpUnicastRtpsPduCollection

The DdsDomainParticipantDpUnicastRtpsPduCollection

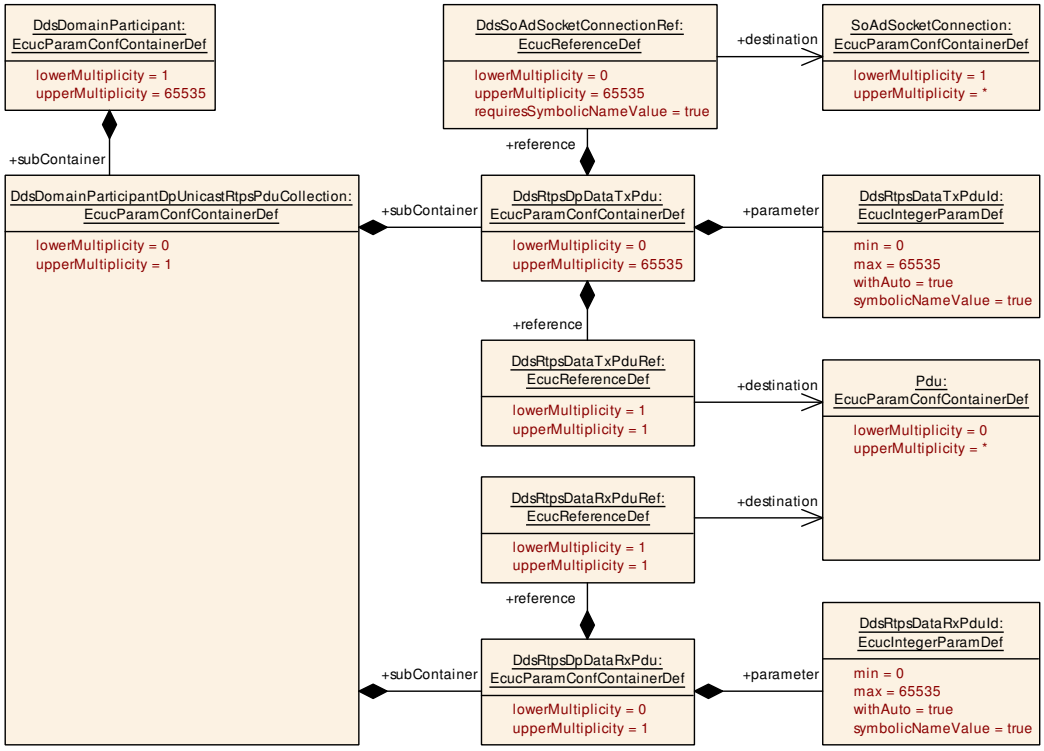


Figure 10.9: DdsDomainParticipantDpUnicastRtpsPduCollection

[ECUC_Dds_00183] Definition of EcucParamConfContainerDef DdsDomainParticipantDpUnicastRtpsPduCollection ⌈

Container Name	DdsDomainParticipantDpUnicastRtpsPduCollection
Parent Container	DdsDomainParticipant
Upstream mapping defined	–
Description	Collection of unicast lower layer PDUs used in service discoved communication towards lower network layers.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsRtpsDpDataRxPdu	0..1	The unicast reception lower layer pdu used to receive service discoved data from the lower network layer to DDS itself.
DdsRtpsDpDataTxPdu	0..65535	The unicast transmission lower layer pdu used to transmit service discoverd data from DDS to lower network layer.

⌋

DdsRtpsDpDataRxPdu

[ECUC_Dds_00187] Definition of EcucParamConfContainerDef DdsRtpsDpDataRxPdu ⌈

Container Name	DdsRtpsDpDataRxPdu	
Parent Container	DdsDomainParticipantDpUnicastRtpsPduCollection	
Upstream mapping defined	–	
Description	The unicast reception lower layer pdu used to receive service discoved data from the lower network layer to DDS itself.	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRtpsDataRxPduId	1	[ECUC_Dds_00149]
DdsRtpsDataRxPduRef	1	[ECUC_Dds_00150]

No Included Containers

⌋

For parameter table [ECUC_Dds_00149] DdsRtpsDataRxPduId, see definition below container DdsRtpsDataRxPdu.

For parameter table [ECUC_Dds_00150] DdsRtpsDataRxPduRef, see definition below container DdsRtpsDataRxPdu.

DdsRtpsDpDataTxPdu

[ECUC_Dds_00184] Definition of EcucParamConfContainerDef DdsRtpsDpDataTxPdu ⌈

Container Name	DdsRtpsDpDataTxPdu	
Parent Container	DdsDomainParticipantDpUnicastRtpsPduCollection	
Upstream mapping defined	–	
Description	The unicast transmission lower layer pdu used to transmit service discoverd data from DDS to lower network layer.	
Multiplicity	0..65535	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRtpsDataTxPduId	1	[ECUC_Dds_00146]
DdsRtpsDataTxPduRef	1	[ECUC_Dds_00147]
DdsSoAdSocketConnectionRef	0..65535	[ECUC_Dds_00185]

No Included Containers

⌋

For parameter table [ECUC_Dds_00146] DdsRtpsDataTxPduId, see definition below container DdsRtpsDataTxPdu.

For parameter table [ECUC_Dds_00147] DdsRtpsDataTxPduRef, see definition below container DdsRtpsDataTxPdu.

For parameter table [ECUC_Dds_00185] DdsSoAdSocketConnectionRef, see definition below container DdsRtpsDataTxPdu.

10.2.3.3.1.3 DdsDomainParticipantQoS

[ECUC_Dds_00013] Definition of EcucParamConfContainerDef DdsDomainParticipantQoS ⌈

Container Name	DdsDomainParticipantQoS
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of QoS supported by the Dds Domain Participant.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsEntityFactory	0..1	If present, this container indicates that Dds ENTITY_FACTORY QoS is supported by this DomainParticipant.
DdsUserData	0..1	If present, this container indicates that Dds USER_DATA QoS is supported by this DomainParticipant.

⌋

DdsUserData For description of this subcontainer, please refer to Chapter 10.2.3.6.1

DdsEntityFactory For description of this subcontainer, please refer to Chapter 10.2.3.6.20

10.2.3.3.1.4 DdsDomainParticipantCryptoInfo

In the picture below, the UML diagram of DdsDomainParticipantCryptoInfo container is shown

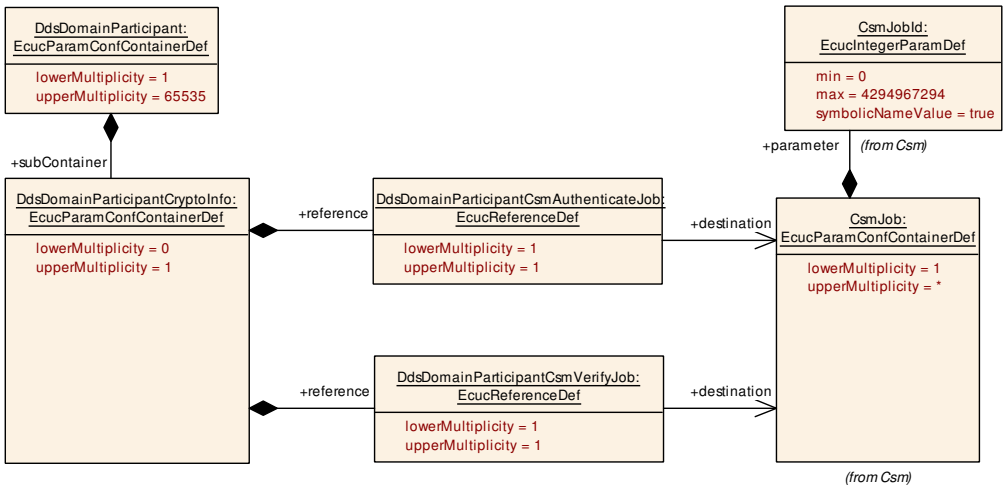


Figure 10.10: Dds DomainParticipant Crypto Info

[ECUC_Dds_00015] Definition of EcucParamConfContainerDef DdsDomainParticipantCryptoInfo ⌈

Container Name	DdsDomainParticipantCryptoInfo
Parent Container	DdsDomainParticipant
Upstream mapping defined	–
Description	This container contains the configuration of the Crypto service to be used by Entities belonging to this DomainParticipant. If it is not present, it means that not security mechanism is supported.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDomainParticipantCsmAuthenticateJob	1	[ECUC_Dds_00020]
DdsDomainParticipantCsmVerifyJob	1	[ECUC_Dds_00021]

No Included Containers

⌋

[ECUC_Dds_00020] Definition of EcucReferenceDef DdsDomainParticipantCsmAuthenticateJob ⌈

Parameter Name	DdsDomainParticipantCsmAuthenticateJob
Parent Container	DdsDomainParticipantCryptoInfo
Upstream mapping defined	–
Description	The reference to the CSM job to be used to authenticate data.
Multiplicity	1
Type	Reference to CsmJob

▽

△

Pre-Compile only	true

⌋

[ECUC_Dds_00021] Definition of EcucReferenceDef DdsDomainParticipantCsmVerifyJob ⌈

Parameter Name	DdsDomainParticipantCsmVerifyJob
Parent Container	DdsDomainParticipantCryptoInfo
Upstream mapping defined	–
Description	The reference to the CSM job to be used to verify data.
Multiplicity	1
Type	Reference to CsmJob
Pre-Compile only	true

⌋

10.2.3.3.1.5 DdsDataSender

The `DdsDataSender` is used to model a DdsWriter linked to a Sender/Receiver interface.

[ECUC_Dds_00157] Definition of EcucParamConfContainerDef DdsDataSender ⌈

Container Name	DdsDataSender
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Container useful to usability: it defines a DDS DataWriter linked to a Sender/Receiver Interface.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsSenderDataWriterRef	1	[ECUC_Dds_00158]

No Included Containers

⌋

[ECUC_Dds_00158] Definition of EcucReferenceDef DdsSenderDataWriterRef ⌈

Parameter Name	DdsSenderDataWriterRef
Parent Container	DdsDataSender
Upstream mapping defined	–

▽

△

Description	Reference to the dataWriter used by the sender
Multiplicity	1
Type	Reference to DdsDataWriter
Pre-Compile only	true

⌋

10.2.3.3.1.6 DdsDataReceiver

The `DdsDataReceiver` is used to model a DdsReader linked to a Sender/Receiver interface.

[ECUC_Dds_00159] Definition of EcucParamConfContainerDef DdsDataReceiver
⌈

Container Name	DdsDataReceiver
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Container useful to usability: it defines a DDS DataReader linked to a Sender/Receiver Interface.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsReceiverDataReaderRef	1	[ECUC_Dds_00160]

No Included Containers

⌋

[ECUC_Dds_00160] Definition of EcucReferenceDef DdsReceiverDataReaderRef
⌈

Parameter Name	DdsReceiverDataReaderRef
Parent Container	DdsDataReceiver
Upstream mapping defined	–
Description	Reference to the dataReader used by the receiver
Multiplicity	1
Type	Reference to DdsDataReader
Pre-Compile only	true

⌋

10.2.3.3.1.7 DdsPublisher

In the picture below, the UML diagram of DdsPublisher container is shown

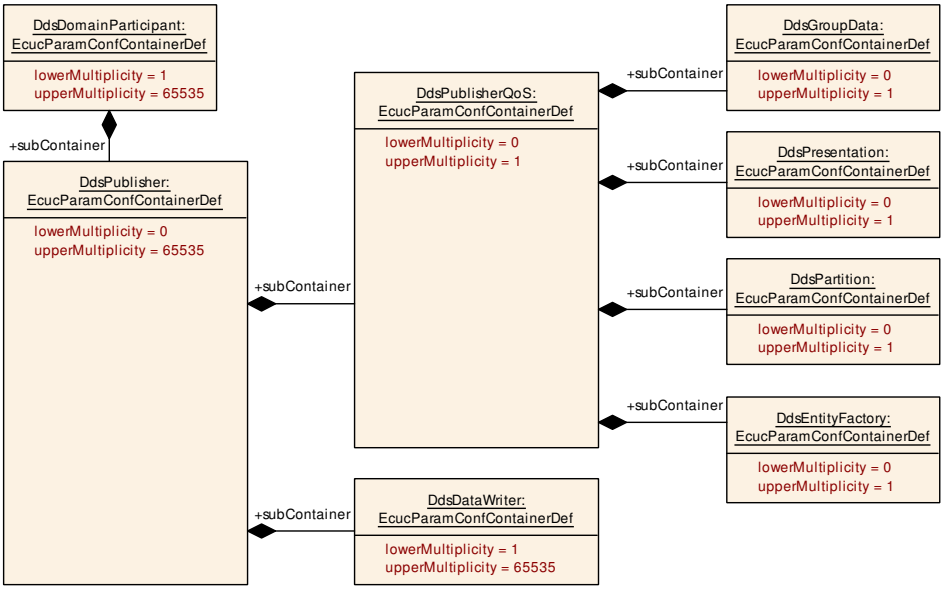


Figure 10.11: Dds Publisher

[ECUC_Dds_00016] Definition of EcucParamConfContainerDef DdsPublisher ⌈

Container Name	DdsPublisher
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of one Publisher.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDataWriter	1..65535	This container represents the configuration of one data writer. One publisher can refer to one or more writer, but a writer can belong to one single publisher.
DdsPublisherQoS	0..1	This container represents the configuration of QoS Profiles related to the current Dds Publisher.

⌋

DdsPublisherQoS

[ECUC_Dds_00022] Definition of EcucParamConfContainerDef DdsPublisherQoS ⌈

Container Name	DdsPublisherQoS
Parent Container	DdsPublisher
Upstream mapping defined	–
Description	This container represents the configuration of QoS Profiles related to the current Dds Publisher.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
DdsEntityFactory	0..1	If present, this container indicates that Dds ENTITY_FACTORY QoS is supported.
DdsGroupData	0..1	If present, this container indicates that Dds GROUP_DATA QoS is supported.
DdsPartition	0..1	If present, this container indicates that Dds PARTITION QoS is supported.
DdsPresentation	0..1	If present, this container indicates that Dds PRESENTATION QoS is supported.

⌋

DdsGroupData For description of this subcontainer, please refer to Chapter 10.2.3.6.3

DdsPresentation For description of this subcontainer, please refer to Chapter 10.2.3.6.6

DdsPartition For description of this subcontainer, please refer to Chapter 10.2.3.6.13

DdsEntityFactory For description of this subcontainer, please refer to Chapter 10.2.3.6.20

DdsDataWriter

In the picture below, the UML diagram of `DdsDataWriter` container is shown

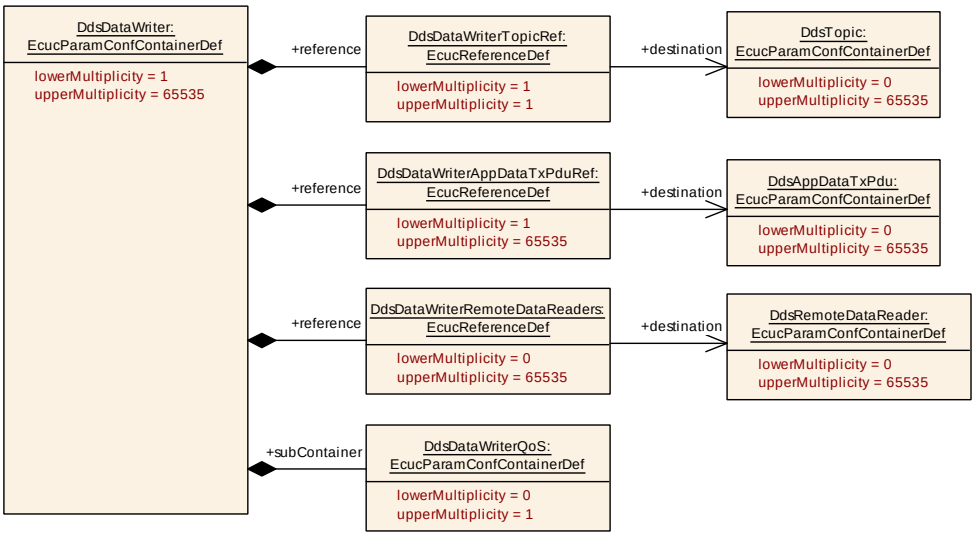


Figure 10.12: Dds DataWriter

[ECUC_Dds_00023] Definition of EcucParamConfContainerDef DdsDataWriter ⌈

Container Name	DdsDataWriter
Parent Container	DdsPublisher
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of one data writer. One publisher can refer to one or more writer, but a writer can belong to one single publisher.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataWriterAppDataTxPduRef	1..65535	[ECUC_Dds_00139]
DdsDataWriterRemoteDataReaders	0..65535	[ECUC_Dds_00140]
DdsDataWriterTopicRef	1	[ECUC_Dds_00029]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDataWriterQoS	0..1	This container represents the configuration of QoS Profiles related to the current DdsDataWriter.

⌋

[ECUC_Dds_00139] Definition of EcucReferenceDef DdsDataWriterAppDataTx PduRef ⌈

Parameter Name	DdsDataWriterAppDataTxPduRef
Parent Container	DdsDataWriter
Upstream mapping defined	–

▽

△

Description	This reference refers to the DdsAppDataTxPdu which is used by the upper layer of the Dds module to transfer data which is requested to be transmitted on the network as DDS serialized data.
Multiplicity	1..65535
Type	Reference to DdsAppDataTxPdu
Pre-Compile only	true

⌋

[ECUC_Dds_00140] Definition of EcucReferenceDef DdsDataWriterRemoteDataReaders ⌈

Parameter Name	DdsDataWriterRemoteDataReaders
Parent Container	DdsDataWriter
Upstream mapping defined	–
Description	Reference to the remote DdsDataReaders that the current DdsDataWriters has to send data to. For each remote DdsDataReader configured, the local DdsDataWriter shall send data by using the DdsDataReaderRemotePdu configured for the given remote DataReader.
Multiplicity	0..65535
Type	Reference to DdsRemoteDataReader
Pre-Compile only	true

⌋

[ECUC_Dds_00029] Definition of EcucReferenceDef DdsDataWriterTopicRef ⌈

Parameter Name	DdsDataWriterTopicRef
Parent Container	DdsDataWriter
Upstream mapping defined	–
Description	This reference selects the Topic on which the current Dds Writer wants to publish.
Multiplicity	1
Type	Reference to DdsTopic
Pre-Compile only	true

⌋

DdsDataWriterQoS

[ECUC_Dds_00028] Definition of EcucParamConfContainerDef DdsDataWriterQoS ⌈

Container Name	DdsDataWriterQoS
Parent Container	DdsDataWriter, DdsRemoteDataWriter
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of QoS Profiles related to the current Dds DataWriter.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDeadline	0..1	If present, this container indicates that Dds DEADLINE QoS is supported.
DdsDestinationOrder	0..1	If present, this container indicates that Dds DESTINATION_ ORDER QoS is supported.
DdsDurability	0..1	If present, this container indicates that Dds DURABILITY QoS is supported.
DdsDurabilityService	0..1	If present, this container indicates that Dds DURABILITY_ SERVICE QoS is supported.
DdsHistory	0..1	If present, this container indicates that Dds HISTORY QoS is supported.
DdsLatencyBudget	0..1	If present, this container indicates that Dds LATENCY_BUDGET QoS is supported.
DdsLifespan	0..1	If present, this container indicates that Dds LIFESPAN QoS is supported.
DdsLiveliness	0..1	If present, this container indicates that Dds LIVELINESS QoS is supported.
DdsOwnership	0..1	If present, this container indicates that Dds OWNERSHIP QoS is supported.
DdsOwnershipStrength	0..1	Describes the DDS [2] OWNERSHIP_STRENGTH QoS policy.
DdsReliability	0..1	If present, this container indicates that Dds RELIABILITY QoS is supported.
DdsResourceLimits	0..1	If present, this container indicates that Dds RESOURCE_LIMITS QoS is supported.
DdsTransportPriority	0..1	If present, this container indicates that Dds TRANSPORT_ PRIORITY QoS is supported.
DdsUserData	0..1	If present, this container indicates that Dds USER_DATA QoS is supported.
DdsWriterDataLifecycle	0..1	Describes the DDS [2] WRITER_DATA_LIFECYCLE QoS policy.

⌋

DdsUserData For description of this subcontainer, please refer to Chapter 10.2.3.6.1

DdsDurability For description of this subcontainer, please refer to Chapter 10.2.3.6.4

DdsDurabilityService For description of this subcontainer, please refer to Chapter 10.2.3.6.5

DdsDeadline For description of this subcontainer, please refer to Chapter 10.2.3.6.7

DdsLatencyBudget For description of this subcontainer, please refer to Chapter 10.2.3.6.8

DdsOwnership For description of this subcontainer, please refer to Chapter 10.2.3.6.9

DdsOwnershipStrength For description of this subcontainer, please refer to Chapter 10.2.3.6.10

DdsLiveliness For description of this subcontainer, please refer to Chapter 10.2.3.6.11

DdsReliability For description of this subcontainer, please refer to Chapter 10.2.3.6.14

DdsTransportPriority For description of this subcontainer, please refer to Chapter 10.2.3.6.15

DdsLifespan For description of this subcontainer, please refer to Chapter 10.2.3.6.16

DdsDestinationOrder For description of this subcontainer, please refer to Chapter 10.2.3.6.17

DdsHistory For description of this subcontainer, please refer to Chapter 10.2.3.6.18

DdsResourceLimits For description of this subcontainer, please refer to Chapter 10.2.3.6.19

DdsWriterDataLifecycle For description of this subcontainer, please refer to Chapter 10.2.3.6.21

10.2.3.3.1.8 DdsSubscriber

In the picture below, the UML diagram of DdsSubscriber container is shown

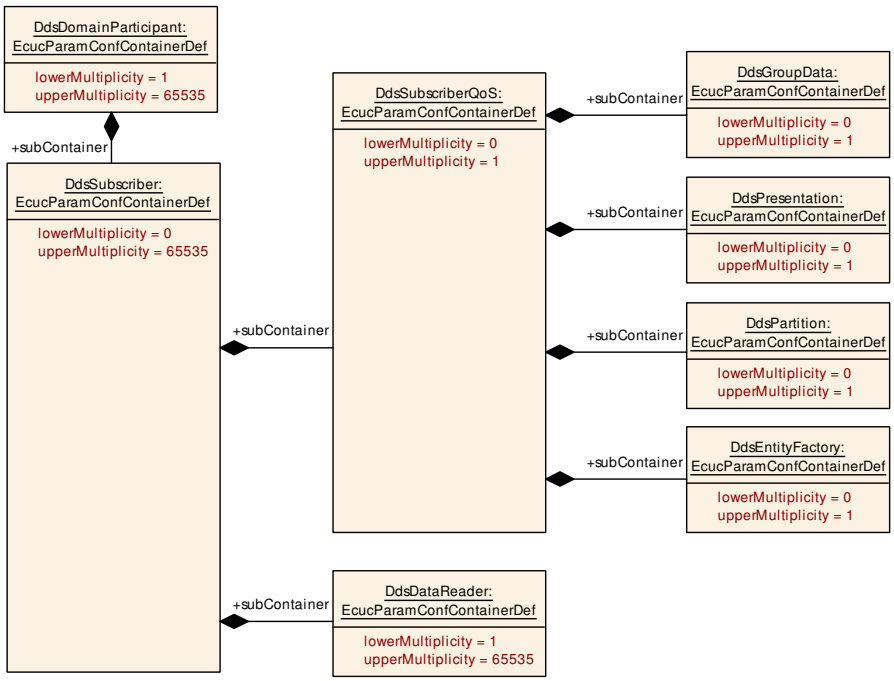


Figure 10.13: Dds Subscriber

[ECUC_Dds_00017] Definition of EcucParamConfContainerDef DdsSubscriber ⌈

Container Name	DdsSubscriber
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of a Subscriber.

▽

△

Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDataReader	1..65535	This container represents the configuration of one data reader. One subscriber can refer to one or more readers, but a reader can belong to one single subscriber.
DdsSubscriberQoS	0..1	This container represents the configuration of QoS Profiles related to the current Dds Subscriber.

⌋

DdsSubscriberQoS

[ECUC_Dds_00074] Definition of EcucParamConfContainerDef DdsSubscriberQoS ⌈

Container Name	DdsSubscriberQoS
Parent Container	DdsSubscriber
Upstream mapping defined	–
Description	This container represents the configuration of QoS Profiles related to the current Dds Subscriber.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsEntityFactory	0..1	If present, this container indicates that Dds ENTITY_FACTORY QoS is supported.
DdsGroupData	0..1	If present, this container indicates that Dds GROUP_DATA QoS is supported.
DdsPartition	0..1	If present, this container indicates that Dds PARTITION QoS is supported.
DdsPresentation	0..1	If present, this container indicates that Dds PRESENTATION Qo S is supported.

⌋

DdsGroupData For description of this subcontainer, please refer to Chapter 10.2.3.6.3

DdsPresentation For description of this subcontainer, please refer to Chapter 10.2.3.6.6

DdsPartition For description of this subcontainer, please refer to Chapter 10.2.3.6.13

DdsEntityFactory For description of this subcontainer, please refer to Chapter 10.2.3.6.20

DdsDataReader

In the picture below, the UML diagram of DdsDataReader container is shown

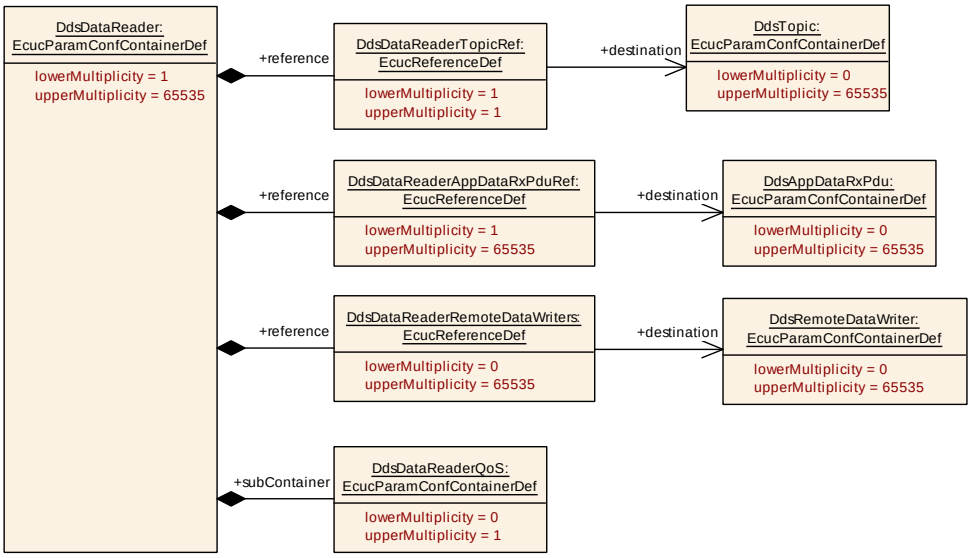


Figure 10.14: Dds DataReader

[ECUC_Dds_00075] Definition of EcucParamConfContainerDef DdsDataReader ⌈

Container Name	DdsDataReader
Parent Container	DdsSubscriber
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of one data reader. One subscriber can refer to one or more readers, but a reader can belong to one single subscriber.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataReaderAppDataRxPduRef	1..65535	[ECUC_Dds_00141]
DdsDataReaderRemoteDataWriters	0..65535	[ECUC_Dds_00142]
DdsDataReaderTopicRef	1	[ECUC_Dds_00076]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDataReaderQoS	0..1	This container represents the configuration of QoS Profiles related to the current DdsDataReader.

⌋

[ECUC_Dds_00141] Definition of EcucReferenceDef DdsDataReaderAppDataRxPduRef ⌈

Parameter Name	DdsDataReaderAppDataRxPduRef
Parent Container	DdsDataReader
Upstream mapping defined	–
Description	This reference refers to the DdsAppDataRxPdu which is used by Dds module to forward de-serialized DDS data to the upper layer of the DDS module.
Multiplicity	1..65535
Type	Reference to DdsAppDataRxPdu
Pre-Compile only	true

⌋

[ECUC_Dds_00142] Definition of EcucReferenceDef DdsDataReaderRemoteDataWriters ⌈

Parameter Name	DdsDataReaderRemoteDataWriters
Parent Container	DdsDataReader
Upstream mapping defined	–
Description	Reference to the remote DdsDataWriters from which the current DdsDataReaders would like to receive data. Data from a remote writer is received by using the DdsRemoteDataWriterPdu referred by the given DdsRemoteDataWriter.
Multiplicity	0..65535
Type	Reference to DdsRemoteDataWriter
Pre-Compile only	true

⌋

[ECUC_Dds_00076] Definition of EcucReferenceDef DdsDataReaderTopicRef ⌈

Parameter Name	DdsDataReaderTopicRef
Parent Container	DdsDataReader
Upstream mapping defined	–
Description	This reference selects the Topic on which the current Dds Reader wants to receive.
Multiplicity	1
Type	Reference to DdsTopic
Pre-Compile only	true

⌋

DdsDataReaderQoS

[ECUC_Dds_00079] Definition of EcucParamConfContainerDef DdsDataReaderQoS ⌈

Container Name	DdsDataReaderQoS
Parent Container	DdsDataReader, DdsRemoteDataReader
Upstream mapping defined	[13, CP_TPS_SystemTemplate]

▽

△

Description	This container represents the configuration of QoS Profiles related to the current Dds DataReader.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
DdsDeadline	0..1	If present, this container indicates that Dds DEADLINE QoS is supported.
DdsDestinationOrder	0..1	If present, this container indicates that Dds DESTINATION_ ORDER QoS is supported.
DdsHistory	0..1	If present, this container indicates that Dds HISTORY QoS is supported.
DdsLatencyBudget	0..1	If present, this container indicates that Dds LATENCY_BUDGET QoS is supported.
DdsLiveliness	0..1	If present, this container indicates that Dds LIVELINESS QoS is supported.
DdsOwnership	0..1	If present, this container indicates that Dds OWNERSHIP QoS is supported.
DdsReaderDataLifecycle	0..1	Describes the DDS [2] READER_DATA_LIFECYCLE QoS policy.
DdsReliability	0..1	If present, this container indicates that Dds RELIABILITY QoS is supported.
DdsResourceLimits	0..1	If present, this container indicates that Dds RESOURCE_LIMITS QoS is supported.
DdsTimeBasedFilter	0..1	Describes the DDS [2] TIME_BASED_FILTER QoS policy.
DdsUserData	0..1	If present, this container indicates that Dds USER_DATA QoS is supported.

⌋

DdsUserData For description of this subcontainer, please refer to Chapter 10.2.3.6.1

DdsDurability For description of this subcontainer, please refer to Chapter 10.2.3.6.4

DdsDeadline For description of this subcontainer, please refer to Chapter 10.2.3.6.7

DdsLatencyBudget For description of this subcontainer, please refer to Chapter 10.2.3.6.8

DdsOwnership For description of this subcontainer, please refer to Chapter 10.2.3.6.9

DdsLiveliness For description of this subcontainer, please refer to Chapter 10.2.3.6.11

DdsTimeBasedFilter For description of this subcontainer, please refer to Chapter 10.2.3.6.12

DdsReliability For description of this subcontainer, please refer to Chapter 10.2.3.6.14

DdsDestinationOrder For description of this subcontainer, please refer to Chapter 10.2.3.6.17

DdsHistory For description of this subcontainer, please refer to Chapter 10.2.3.6.18

DdsResourceLimits For description of this subcontainer, please refer to Chapter 10.2.3.6.19

DdsReaderDataLifecycle For description of this subcontainer, please refer to Chapter 10.2.3.6.22

10.2.3.3.1.9 DdsTopic

In the picture below, the UML diagram of DdsTopic container is shown

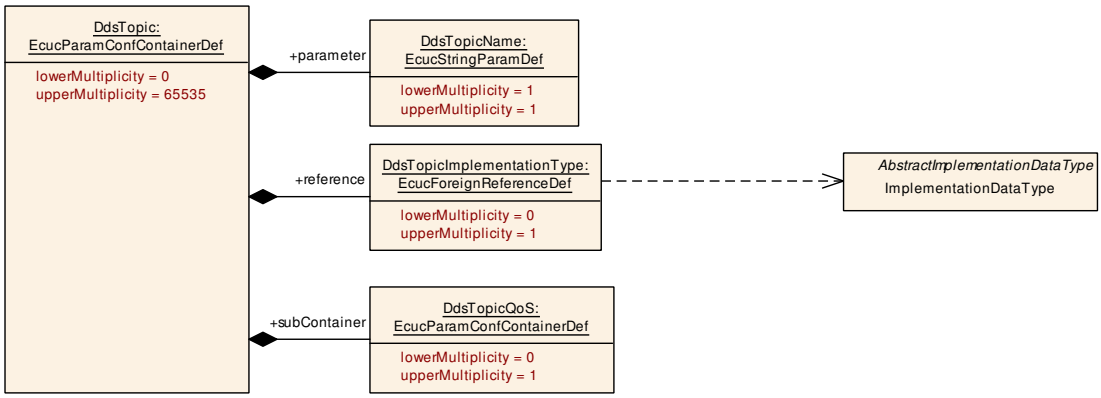


Figure 10.15: Dds Topic

[ECUC_Dds_00018] Definition of EcucParamConfContainerDef DdsTopic ⌈

Container Name	DdsTopic
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of one Topic.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsTopicName	1	[ECUC_Dds_00103]
DdsTopicImplementationType	0..1	[ECUC_Dds_00104]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsTopicQoS	0..1	This container contains the configuration of the QoS supported by the DdsTopic

⌋

[ECUC_Dds_00103] Definition of EcucStringParamDef DdsTopicName ⌈

Parameter Name	DdsTopicName
Parent Container	DdsTopic
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Identifies name of the Topic. Communication between publishers and subscribers is based on the topic name. Dependency: inter-ECU - this value shall be shared between configurations.
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dds_00104] Definition of EcucForeignReferenceDef DdsTopicImplementationType ⌈

Parameter Name	DdsTopicImplementationType
Parent Container	DdsTopic
Upstream mapping defined	–
Description	This reference selects the ImplementationDataType the topic is related. A Topic is used to publish a well-defined data type, described by the referenced ImplementationData Type. Note: if the Topic is related to a SenderReceiver communication, the reference to the ImplementationDataType shall exist
Multiplicity	0..1
Type	Foreign reference to IMPLEMENTATION-DATA-TYPE
Pre-Compile only	true

⌋

DdsTopicQos

[ECUC_Dds_00102] Definition of EcucParamConfContainerDef DdsTopicQoS ⌈

Container Name	DdsTopicQoS
Parent Container	DdsTopic
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container contains the configuration of the QoS supported by the DdsTopic
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDeadline	0..1	If present, this container indicates that Dds DEADLINE QoS is supported.
DdsDestinationOrder	0..1	If present, this container indicates that Dds DESTINATION_ ORDER QoS is supported.
DdsDurability	0..1	If present, this container indicates that Dds DURABILITY QoS is supported.
DdsDurabilityService	0..1	If present, this container indicates that Dds DURABILITY_ SERVICE QoS is supported.
DdsHistory	0..1	If present, this container indicates that Dds HISTORY QoS is supported.
DdsLatencyBudget	0..1	If present, this container indicates that Dds LATENCY_BUDGET QoS is supported.
DdsLifespan	0..1	If present, this container indicates that Dds LIFESPAN QoS is supported.
DdsLiveliness	0..1	If present, this container indicates that Dds LIVELINESS QoS is supported.
DdsOwnership	0..1	If present, this container indicates that Dds OWNERSHIP QoS is supported.
DdsReliability	0..1	If present, this container indicates that Dds RELIABILITY QoS is supported.
DdsResourceLimits	0..1	If present, this container indicates that Dds RESOURCE_LIMITS QoS is supported.
DdsTopicData	0..1	Describes the DDS [2] TOPIC_DATA QoS policy.
DdsTransportPriority	0..1	If present, this container indicates that Dds TRANSPORT_ PRIORITY QoS is supported.

⌋

DdsTopicData For description of this subcontainer, please refer to Chapter 10.2.3.6.2

DdsDurability For description of this subcontainer, please refer to Chapter 10.2.3.6.4

DdsDurabilityService For description of this subcontainer, please refer to Chapter 10.2.3.6.5

DdsDeadline For description of this subcontainer, please refer to Chapter 10.2.3.6.7

DdsLatencyBudget For description of this subcontainer, please refer to Chapter 10.2.3.6.8

DdsOwnership For description of this subcontainer, please refer to Chapter 10.2.3.6.9

DdsLiveliness For description of this subcontainer, please refer to Chapter 10.2.3.6.11

DdsReliability For description of this subcontainer, please refer to Chapter 10.2.3.6.14

DdsTransportPriority For description of this subcontainer, please refer to Chapter 10.2.3.6.15

DdsLifespan For description of this subcontainer, please refer to Chapter 10.2.3.6.16

DdsDestinationOrder For description of this subcontainer, please refer to Chapter 10.2.3.6.17

DdsHistory For description of this subcontainer, please refer to Chapter 10.2.3.6.18

DdsResourceLimits For description of this subcontainer, please refer to Chapter 10.2.3.6.19

10.2.3.3.1.10 DdsRemoteDomainParticipantCollection

The `Dds` shall support complete static configuration of remotes DdsDomainParticipants of each local DdsDomainParticipant, in order to be able to correctly work also in environments where Dynamic discovery is not supported. The `DdsRemoteDomainParticipant` is used to statically configure remote DdsDomainParticipants.

In the picture below, the UML diagram of DdsRemoteDomainParticipant template is shown:

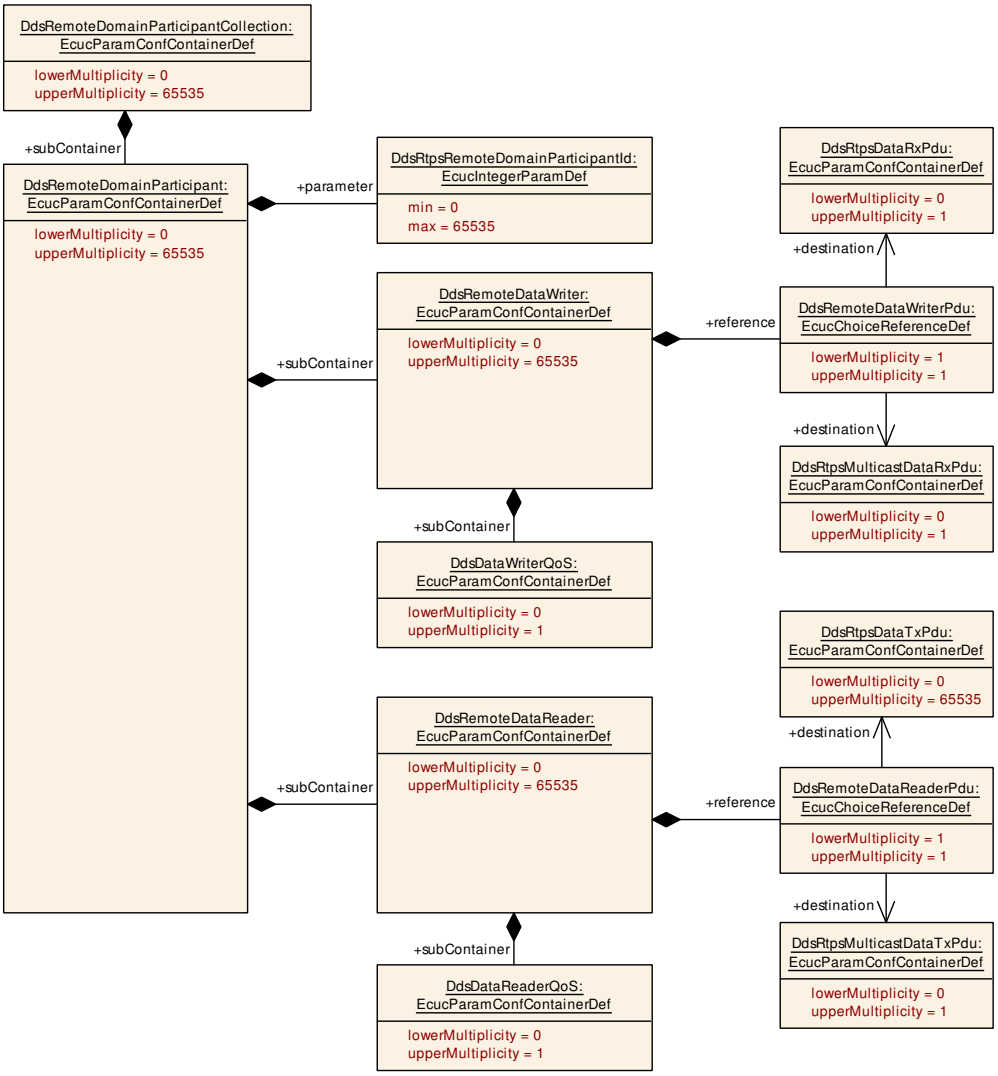


Figure 10.16: DdsRemoteDomainParticipant

[ECUC_Dds_00182] Definition of EcucParamConfContainerDef DdsRemoteDomainParticipantCollection ⌈

Container Name	DdsRemoteDomainParticipantCollection
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Collection of Remote Domain Participants.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsRemoteDomainParticipant	0..65535	Static configuration of remote endpoints. This container contains information about reachability and QoS parameters of remote endpoints.

⌋

DdsRemoteDomainParticipant

[ECUC_Dds_00161] Definition of EcucParamConfContainerDef DdsRemoteDomainParticipant ⌈

Container Name	DdsRemoteDomainParticipant
Parent Container	DdsRemoteDomainParticipantCollection
Upstream mapping defined	–
Description	Static configuration of remote endpoints. This container contains information about reachability and QoS parameters of remote endpoints.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRtpsRemoteDomainParticipantId	1	[ECUC_Dds_00162]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsRemoteDataReader	0..65535	The configuration of a specific remote DataReader.
DdsRemoteDataWriter	0..65535	The configuration of a specific remote DataWriter.

⌋

[ECUC_Dds_00162] Definition of EcucIntegerParamDef DdsRtpsRemoteDomainParticipantId ⌈

Parameter Name	DdsRtpsRemoteDomainParticipantId
Parent Container	DdsRemoteDomainParticipant
Upstream mapping defined	–
Description	The DomainParticipant ID of the remote DDS node. Note: please refer to chapter "User Traffic" of Data Distribution Service (DDS), Version 1.4 for details on its usage.
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65535
Default value	–
Pre-Compile only	true

⌋

DdsRemoteDataWriter

The `DdsRemoteDataWriter` container is used to configure remotes DdsDataWriters for a given local DdsDataReader. The **DdsRemoteDataWriterPdu** is the lower layer pdu to be used by the local `DdsDataReader` to receive data from the referred `DdsRemoteDataWriter`.

[ECUC_Dds_00163] Definition of EcucParamConfContainerDef DdsRemoteDataWriter ⌈

Container Name	DdsRemoteDataWriter
Parent Container	DdsRemoteDomainParticipant
Upstream mapping defined	–
Description	The configuration of a specific remote DataWriter.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRemoteDataWriterPdu	1	[ECUC_Dds_00164]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDataWriterQoS	0..1	This container represents the configuration of QoS Profiles related to the current DdsDataWriter.

⌋

[ECUC_Dds_00164] Definition of EcucChoiceReferenceDef DdsRemoteDataWriterPdu ⌈

Parameter Name	DdsRemoteDataWriterPdu
Parent Container	DdsRemoteDataWriter
Upstream mapping defined	–
Description	The Pdu used to receive data from the given remote DataWriter. It could refer both a unicast and a multicast Pdu
Multiplicity	1
Type	Choice reference to [DdsRtpsDataRxPdu, DdsRtpsMulticastDataRxPdu]
Pre-Compile only	true

⌋

DdsRemoteDataReader

The `DdsRemoteDataReader` container is used to configure remotes DdsDataReaders for a given local DdsDataWriter. The **DdsRemoteDataReaderPdu** is the lower layer pdu to be used by the local `DdsDataWriter` to transmit data to the referred `DdsRemoteDataReader`.

[ECUC_Dds_00165] Definition of EcucParamConfContainerDef DdsRemoteDataReader ⌈

Container Name	DdsRemoteDataReader
Parent Container	DdsRemoteDomainParticipant
Upstream mapping defined	–
Description	The configuration of a specific remote DataReader.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsRemoteDataReaderPdu	1	[ECUC_Dds_00166]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsDataReaderQoS	0..1	This container represents the configuration of QoS Profiles related to the current DdsDataReader.

⌋

[ECUC_Dds_00166] Definition of EcucChoiceReferenceDef DdsRemoteDataReaderPdu ⌈

Parameter Name	DdsRemoteDataReaderPdu
Parent Container	DdsRemoteDataReader
Upstream mapping defined	–

▽

△

Description	The Pdu used to transmit data to the given remote DataReader. It could refer both a unicast and a multicast Pdu
Multiplicity	1
Type	Choice reference to [DdsRtpsDataTxPdu, DdsRtpsMulticastDataTxPdu]
Pre-Compile only	true

⌋

10.2.3.3.1.11 DdsProvidedServiceInstance

The DdsProvidedServiceInstance Dds Provided Service instance is the container suitable to collect all kinds of service interface elements by server side.

[ECUC_Dds_00193] Definition of EcucParamConfContainerDef DdsProvidedServiceInstance ⌈

Container Name	DdsProvidedServiceInstance
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents the configuration of a provided service instance at Ecuc level in a concrete implementation on top of DDS.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDiscoveryType	1	[ECUC_Dds_00197]
DdsFieldReplyTopicName	0..1	[ECUC_Dds_00199]
DdsFieldRequestTopicName	0..1	[ECUC_Dds_00198]
DdsMethodReplyTopicName	0..1	[ECUC_Dds_00201]
DdsMethodRequestTopicName	0..1	[ECUC_Dds_00200]
DdsServiceInstanceId	0..1	[ECUC_Dds_00195]
DdsServiceInterfaceId	1	[ECUC_Dds_00194]
DdsServiceMajorVersion	1	[ECUC_Dds_00203]
DdsServiceMinorVersion	1	[ECUC_Dds_00204]
DdsTransportProtocol	1	[ECUC_Dds_00202]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsServiceEventSender	0..65535	This container represents an Event sender relying on DdsData Writer linked to a VariableDataPrototype in a SenderReceiver Interface.
DdsServiceFieldsFieldServer	0..65535	This object collects the configuration needed to handle the Field actions on server side.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsServiceMethodsFireAndForgetMethod	0..65535	This object collects all the configuration needed to handle Methods relying on the Fire And Forget communication paradigms. Role (e.g server v.s side) is defined in the parent object.
DdsServiceMethodsRequestReplyMethod	0..65535	This object collects all the configuration needed to handle Methods relying on the Remote Procedure Call communication paradigms. Role (e.g server v.s side) is defined in the parent object.

⌋

For parameter table [ECUC_Dds_00197] DdsDiscoveryType, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00199] DdsFieldReplyTopicName, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00198] DdsFieldRequestTopicName, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00201] DdsMethodReplyTopicName, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00200] DdsMethodRequestTopicName, see definition below container DdsConsumedServiceInstance.

[ECUC_Dds_00195] Definition of EcucIntegerParamDef DdsServiceInstanceId ⌈

Parameter Name	DdsServiceInstanceId	
Parent Container	DdsProvidedServiceInstance	
Upstream mapping defined	–	
Description	The service instance identifier (This is used by DDS to identify DomainParticipants associated with an instance of the service).	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	–	
Pre-Compile only	true	

⌋

For parameter table [ECUC_Dds_00194] DdsServiceInterfaceId, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00203] DdsServiceMajorVersion, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00204] DdsServiceMinorVersion, see definition below container DdsConsumedServiceInstance.

For parameter table [ECUC_Dds_00202] DdsTransportProtocol, see definition below container DdsConsumedServiceInstance.

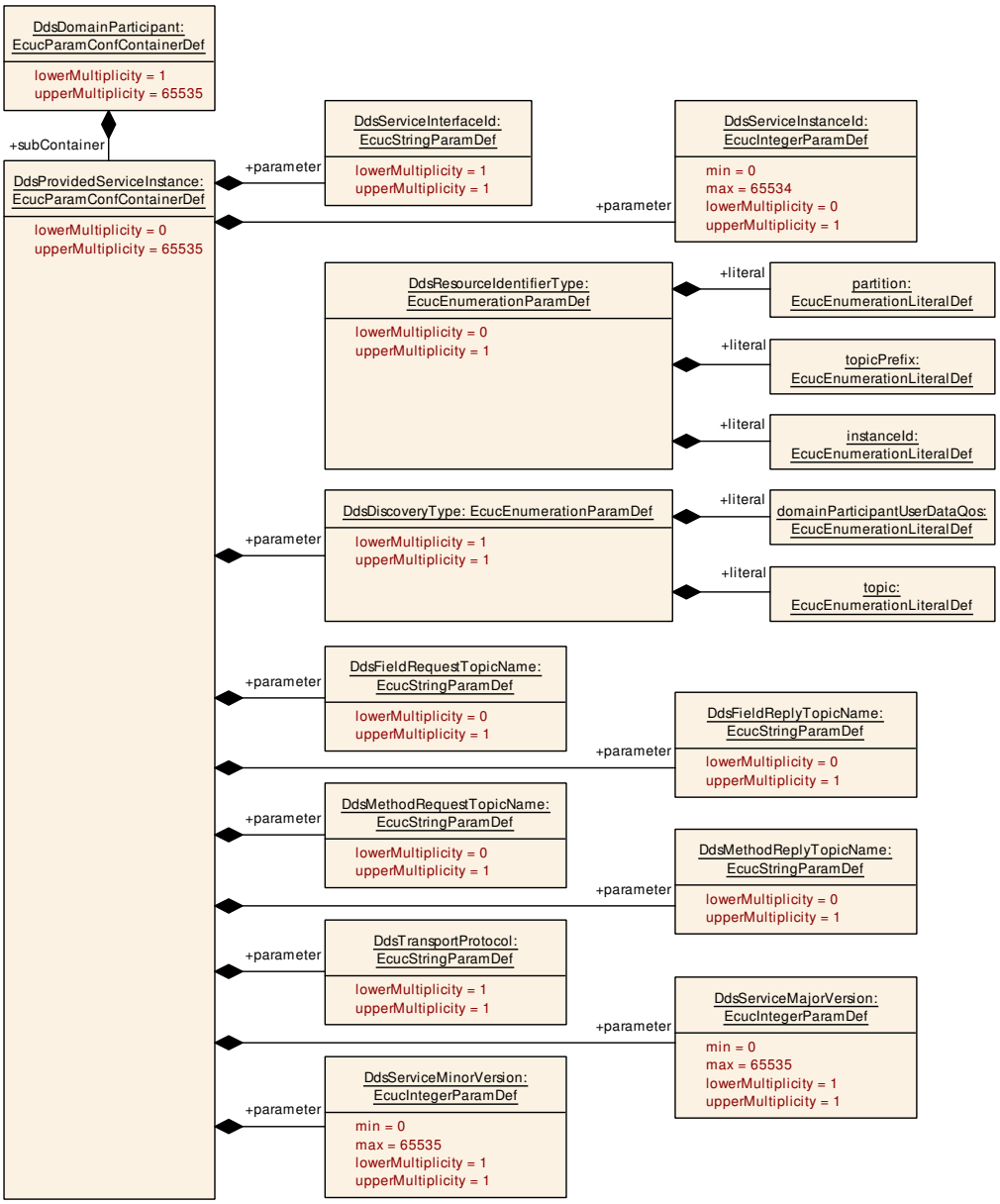


Figure 10.17: DdsProvidedServiceInstance

[CP_SWS_Dds_CONSTR_00970] DdsDomainParticipant **can aggregate only one** DdsProvidedServiceInstance ⌈DdsDomainParticipant can aggregate only one DdsProvidedServiceInstance⌋

[ECUC_Dds_00205] Definition of EcucParamConfContainerDef DdsServiceEvent Sender ⌈

Container Name	DdsServiceEventSender
Parent Container	DdsProvidedServiceInstance
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents an Event sender relying on DdsDataWriter linked to a VariableDataPrototype in a SenderReceiverInterface.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataElementRef	0..1	[ECUC_Dds_00206]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]

No Included Containers

⌋

For parameter table [ECUC_Dds_00206] DdsDataElementRef, see definition below container DdsServiceEventReceiver.

[ECUC_Dds_00207] Definition of EcucReferenceDef DdsDataWriterRef ⌈

Parameter Name	DdsDataWriterRef
Parent Container	DdsServiceEventSender, DdsServiceFieldsFieldClientGetter, DdsServiceFieldsFieldClientSetter, DdsServiceFieldsFieldNotifierSender, DdsServiceFieldsFieldServerGetter, DdsServiceFieldsFieldServerSetter, DdsServiceMethodsFireAndForgetMethod, DdsServiceMethodsRequestReplyMethod
Upstream mapping defined	–
Description	This object represents a reference to a DdsDataWriter. It is the main object modeling the transmission communication path in a Dds service instance.
Multiplicity	0..1
Type	Reference to DdsDataWriter
Pre-Compile only	true

⌋

[CP_SWS_Dds_CONSTR_00950] Event sender DdsDataElementRef ⌈if `DdsServiceEventSender` exists then `DdsServiceEventSender.DdsDataElementRef` shall be defined.⌋

[CP_SWS_Dds_CONSTR_00952] Event sender DdsDataWriterRef ⌈If `DdsServiceEventSender` exists then `DdsServiceEventSender.DdsDataWriterRef` shall be defined.⌋

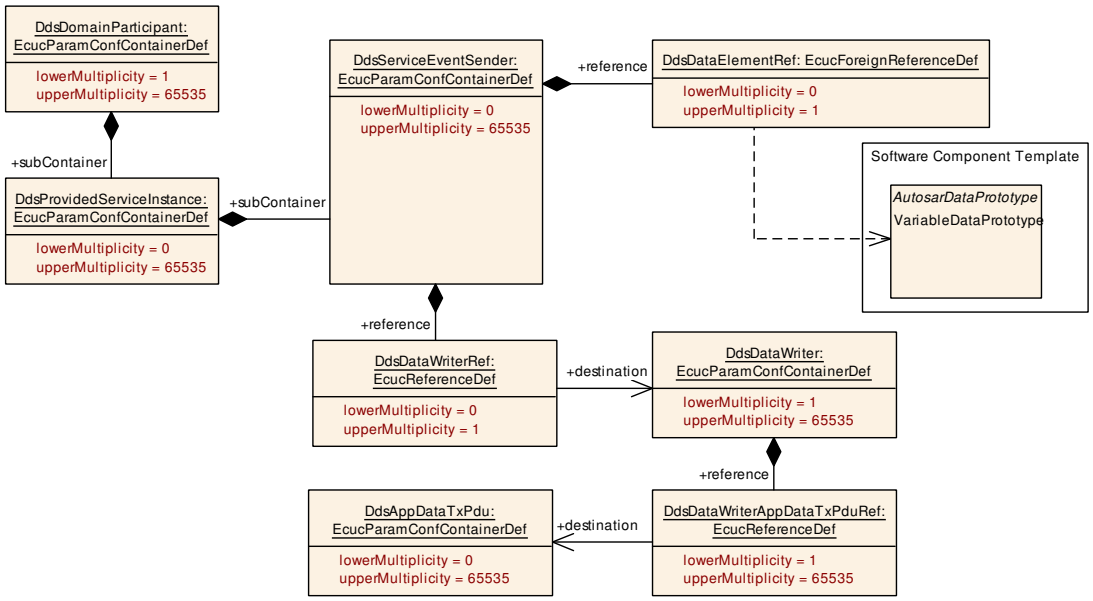


Figure 10.18: DdsServiceEventSender

[ECUC_Dds_00215] Definition of EcucParamConfContainerDef DdsServiceMethodsRequestReplyMethod ⌈

Container Name	DdsServiceMethodsRequestReplyMethod
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects all the configuration needed to handle Methods relying on the Remote Procedure Call communication paradigms. Role (e.g server v.s side) is defined in the parent object.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataReaderRef	0..1	[ECUC_Dds_00211]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]
DdsOperationRef	0..1	[ECUC_Dds_00216]

No Included Containers

⌋

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

[ECUC_Dds_00216] Definition of EcucForeignReferenceDef DdsOperationRef ⌈

Parameter Name	DdsOperationRef
Parent Container	DdsServiceMethodsRequestReplyMethod
Upstream mapping defined	–
Description	This object represents a foreign reference to a ClientServerOperation.
Multiplicity	0..1
Type	Foreign reference to CLIENT-SERVER-OPERATION
Pre-Compile only	true

⌋

[ECUC_Dds_00217] Definition of EcucParamConfContainerDef DdsServiceMethodsFireAndForgetMethod ⌈

Container Name	DdsServiceMethodsFireAndForgetMethod
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects all the configuration needed to handle Methods relying on the Fire And Forget communication paradigms. Role (e.g server v.s side) is defined in the parent object.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataElementRef	0..1	[ECUC_Dds_00206]
DdsDataReaderRef	0..1	[ECUC_Dds_00211]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]
DdsTriggerRef	0..1	[ECUC_Dds_00218]

No Included Containers

⌋

For parameter table [ECUC_Dds_00206] DdsDataElementRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

[ECUC_Dds_00218] Definition of EcucForeignReferenceDef DdsTriggerRef ⌈

Parameter Name	DdsTriggerRef
Parent Container	DdsServiceMethodsFireAndForgetMethod
Upstream mapping defined	–
Description	This object represents a foreign reference to a Trigger in a TriggerInterface in case the FireAndForget method has no arguments.
Multiplicity	0..1
Type	Foreign reference to TRIGGER
Pre-Compile only	true

⌋

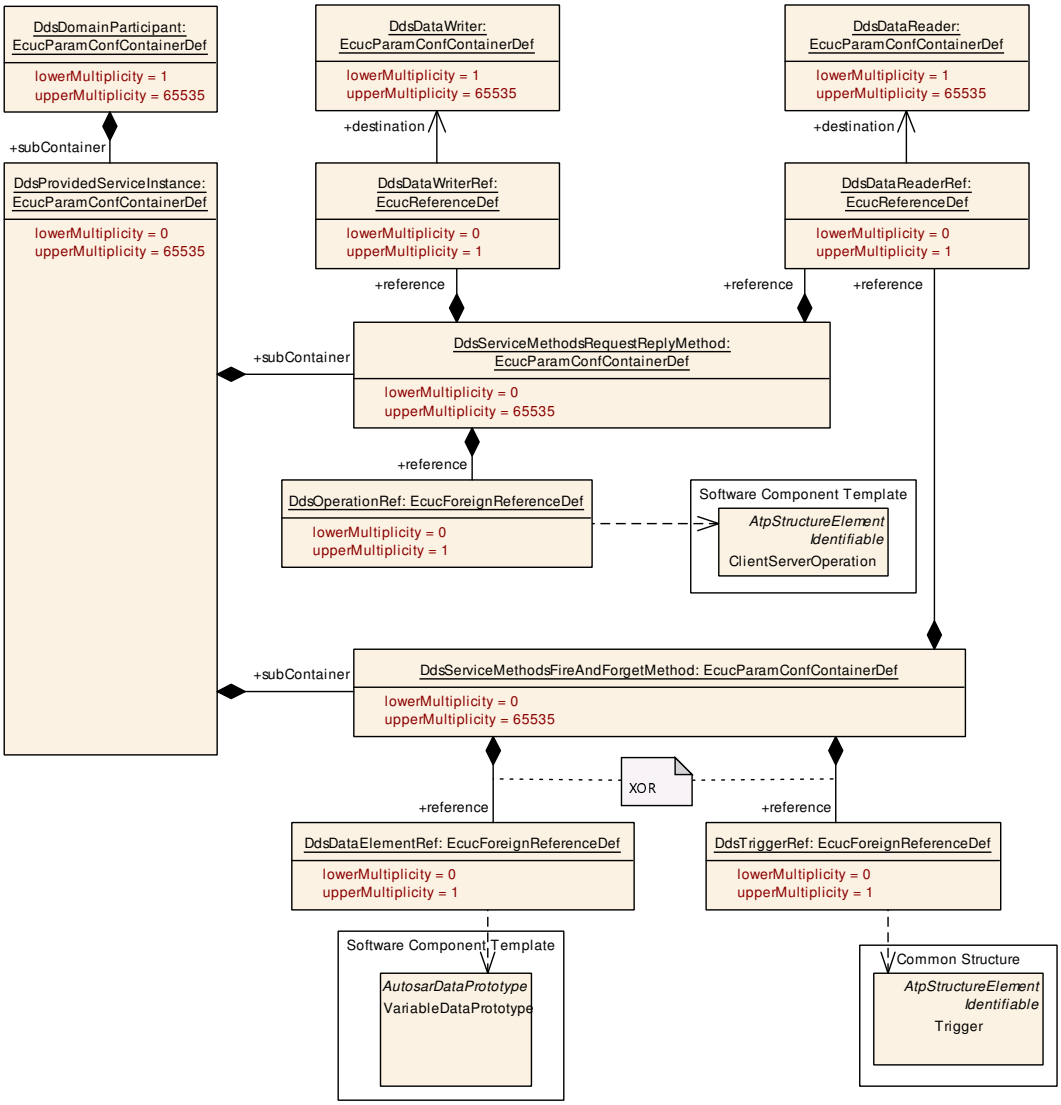


Figure 10.19: DdsServiceMethodsServer

[ECUC_Dds_00208] Definition of EcucParamConfContainerDef DdsServiceFields FieldServer ⌈

Container Name	DdsServiceFieldsFieldServer
Parent Container	DdsProvidedServiceInstance
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle the Field actions on server side.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
DdsServiceFieldsFieldNotifierSender	0..1	This object collects configuration needed to handle all the Fields Notify action on server side.
DdsServiceFieldsFieldServerGetter	0..1	This object collects the configuration needed to handle all the Fields Getter action on server side.
DdsServiceFieldsFieldServerSetter	0..1	This object collects the configuration needed to handle all the Fields Setter action on server side.

⌋

[ECUC_Dds_00209] Definition of EcucParamConfContainerDef DdsServiceFieldsFieldNotifierSender ⌈

Container Name	DdsServiceFieldsFieldNotifierSender
Parent Container	DdsServiceFieldsFieldServer
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects configuration needed to handle all the Fields Notify action on server side.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DdsDataElementRef	0..1	[ECUC_Dds_00206]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]

No Included Containers

⌋

For parameter table [ECUC_Dds_00206] DdsDataElementRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

[CP_SWS_Dds_CONSTR_00954] Field Notifier sender DdsDataElementRef ⌈If `DdsServiceFieldsFieldNotifierSender` exists then `DdsServiceFieldsFieldNotifierSender.DdsDataElementRef` shall be defined.⌋

[CP_SWS_Dds_CONSTR_00958] Field Notifier sender DdsDataWriterRef ⌈If `DdsServiceFieldsFieldNotifierSender` exists then `DdsServiceFieldsFieldNotifierSender.DdsDataWriterRef` shall be defined.⌋

[ECUC_Dds_00210] Definition of EcucParamConfContainerDef DdsServiceFieldsFieldServerGetter ⌈

Container Name	DdsServiceFieldsFieldServerGetter
Parent Container	DdsServiceFieldsFieldServer
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle all the Fields Getter action on server side.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataReaderRef	0..1	[ECUC_Dds_00211]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]
DdsGetOperationRef	0..1	[ECUC_Dds_00212]

No Included Containers

⌋

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

For parameter table [ECUC_Dds_00212] DdsGetOperationRef, see definition below container DdsServiceFieldsFieldClientGetter.

[ECUC_Dds_00214] Definition of EcucParamConfContainerDef DdsServiceFieldsFieldServerSetter ⌈

Container Name	DdsServiceFieldsFieldServerSetter
Parent Container	DdsServiceFieldsFieldServer
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle all the Fields Setter action on server side.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataReaderRef	0..1	[ECUC_Dds_00211]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]
DdsSetOperationRef	0..1	[ECUC_Dds_00213]

No Included Containers

⌋

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

For parameter table [ECUC_Dds_00213] DdsSetOperationRef, see definition below container DdsServiceFieldsFieldClientSetter.

10.2.3.3.1.12 DdsConsumedServiceInstance

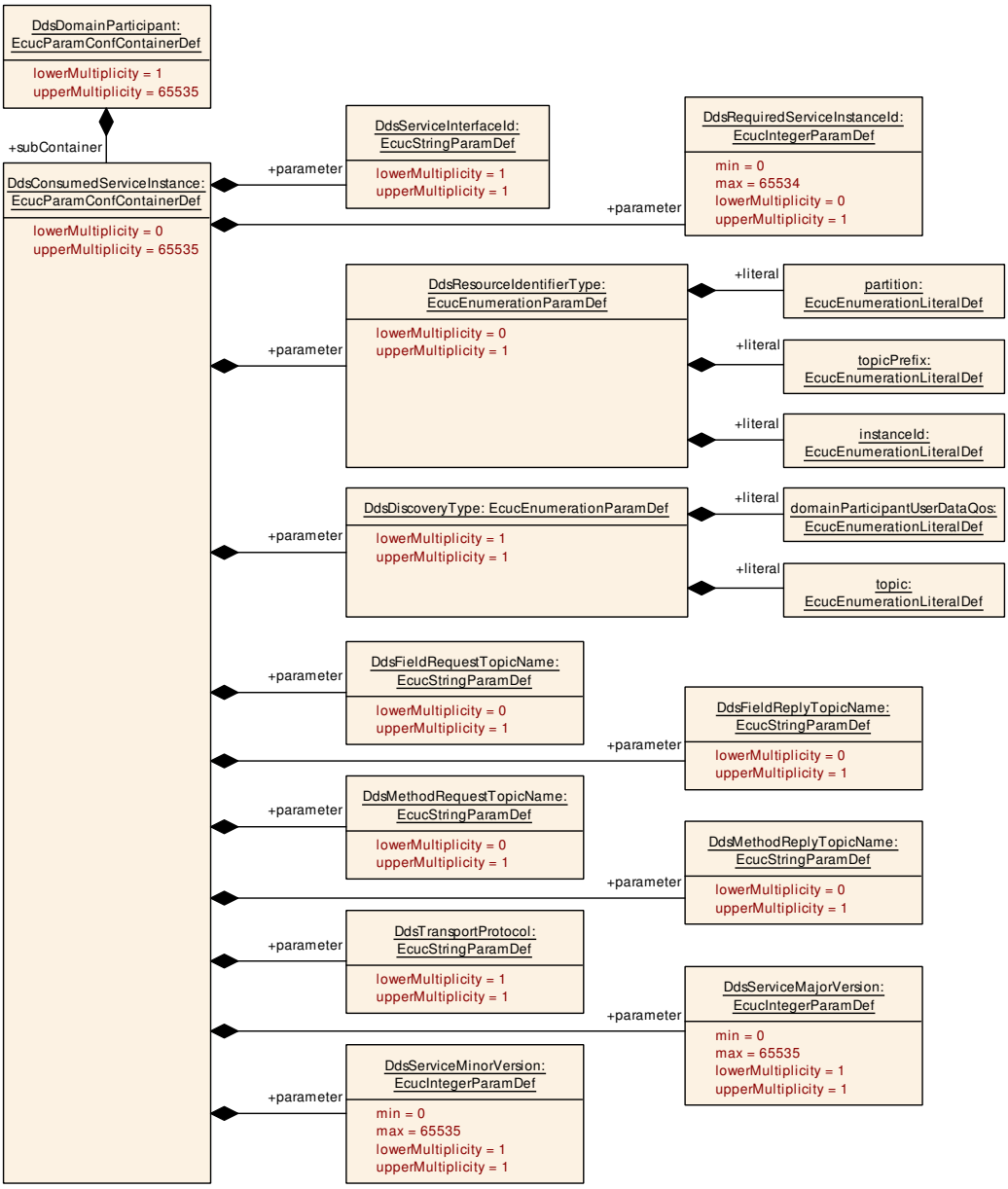


Figure 10.20: DdsConsumedServiceInstance

The `DdsConsumedServiceInstance` is the container suitable to collect all kinds of service interface elements by the client side.

[ECUC_Dds_00219] Definition of EcucParamConfContainerDef DdsConsumed ServiceInstance ⌈

Container Name	DdsConsumedServiceInstance
Parent Container	DdsDomainParticipant
Upstream mapping defined	[13, CP_TPS_SystemTemplate]

▽

△

Description	This container represents the configuration of consumed service instance at Ecuc level in a concrete implementation on top of DDS.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDiscoveryType	1	[ECUC_Dds_00197]
DdsFieldReplyTopicName	0..1	[ECUC_Dds_00199]
DdsFieldRequestTopicName	0..1	[ECUC_Dds_00198]
DdsMethodReplyTopicName	0..1	[ECUC_Dds_00201]
DdsMethodRequestTopicName	0..1	[ECUC_Dds_00200]
DdsRequiredServiceInstanceId	0..1	[ECUC_Dds_00220]
DdsResourceIdentifierType	0..1	[ECUC_Dds_00196]
DdsServiceInterfaceId	1	[ECUC_Dds_00194]
DdsServiceMajorVersion	1	[ECUC_Dds_00203]
DdsServiceMinorVersion	1	[ECUC_Dds_00204]
DdsTransportProtocol	1	[ECUC_Dds_00202]

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsServiceEventReceiver	0..65535	This container represents an Event receiver relying on DdsData Writer linked to a VariableDataPrototype in a SenderReceiver interface.
DdsServiceFieldsFieldClient	0..65535	This object collects the configuration needed to handle all the Fields actions on client side.
DdsServiceMethodsFireAndForgetMethod	0..65535	This object collects all the configuration needed to handle Methods relying on the Fire And Forget communication paradigms. Role (e.g server v.s side) is defined in the parent object.
DdsServiceMethodsRequestReplyMethod	0..65535	This object collects all the configuration needed to handle Methods relying on the Remote Procedure Call communication paradigms. Role (e.g server v.s side) is defined in the parent object.

⌋

[ECUC_Dds_00197] Definition of EcucEnumerationParamDef DdsDiscoveryType

⌈

Parameter Name	DdsDiscoveryType	
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Supported discovery schemes for DDS Service Instances.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	domainParticipantUserDataQos	The USER_DATA QoS policy is used to advertise and discover available Service Instances hosted by each Domain Participant [18].

▽

△

	topic	A purpose-specific Topic is used to convey availability of Service Instances and how to bind against them [18].
Pre-Compile only	true	

⌋

[ECUC_Dds_00199] Definition of EcucStringParamDef DdsFieldReplyTopicName ⌈

Parameter Name	DdsFieldReplyTopicName
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	Name of the DDS Reply Topic associated with the Field. Dependency: inter-ECU - this value shall be shared between configurations.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dds_00198] Definition of EcucStringParamDef DdsFieldRequestTopicName ⌈

Parameter Name	DdsFieldRequestTopicName
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	Name of the DDS Request Topic associated with the Field. Dependency: inter-ECU - this value shall be shared between configurations.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dds_00201] Definition of EcucStringParamDef DdsMethodReplyTopicName ⌈

Parameter Name	DdsMethodReplyTopicName
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	Name of the DDS Reply Topic associated with the Method. Dependency: inter-ECU - this value shall be shared between configurations.
Multiplicity	0..1

▽

△

Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dds_00200] Definition of EcucStringParamDef DdsMethodRequestTopicName ⌈

Parameter Name	DdsMethodRequestTopicName
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	Name of the DDS Request Topic associated with the Method. Dependency: inter-ECU - this value shall be shared between configurations.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dds_00220] Definition of EcucIntegerParamDef DdsRequiredServiceInstanceId ⌈

Parameter Name	DdsRequiredServiceInstanceId
Parent Container	DdsConsumedServiceInstance
Upstream mapping defined	–
Description	This attribute represents the ability to describe the required service instance ID If configured to 0xFFFF (ANY), any instance ID will be accepted.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	0 .. 65534
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00196] Definition of EcucEnumerationParamDef DdsResourceIdentifierType ⌈

Parameter Name	DdsResourceIdentifierType
Parent Container	DdsConsumedServiceInstance
Upstream mapping defined	–
Description	In-band instance identification fields are used to discriminate samples related to specific Service Instances sharing the same DDS Topics.
Multiplicity	0..1

▽

△

Type	EcucEnumerationParamDef	
Range	instanceId	In-band instance identification fields are used to discriminate samples related to specific Service Instances sharing the same DDS Topics [18].
	partition	The DDS PARTITION QoS policy is used to isolate DDS Topics related to specific Service Instances [18].
	topicPrefix	Unique prefixes are assigned to DDS Topics related to specific Service Instances [18].
Pre-Compile only	true	

⌋

[ECUC_Dds_00194] Definition of EcucStringParamDef DdsServiceInterfaceId ⌈

Parameter Name	DdsServiceInterfaceId
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	The service interface identifier.
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dds_00203] Definition of EcucIntegerParamDef DdsServiceMajorVersion ⌈

Parameter Name	DdsServiceMajorVersion	
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance	
Upstream mapping defined	–	
Description	Service Major version.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00204] Definition of EcucIntegerParamDef DdsServiceMinorVersion ⌈

Parameter Name	DdsServiceMinorVersion
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	Service Minor version.

▽

△

Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65535
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00202] Definition of EcucStringParamDef DdsTransportProtocol ⌈

Parameter Name	DdsTransportProtocol
Parent Container	DdsConsumedServiceInstance, DdsProvidedServiceInstance
Upstream mapping defined	–
Description	This attribute defines which Transport Layer Protocol(s) this Instance is intended to use. Dependency: inter-ECU - this value shall be shared between configurations.
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[CP_SWS_Dds_CONSTR_00971] DdsDomainParticipant **can aggregate only one** DdsConsumedServiceInstance ⌈DdsDomainParticipant can aggregate only one DdsConsumedServiceInstance⌋

[ECUC_Dds_00221] Definition of EcucParamConfContainerDef DdsServiceEventReceiver ⌈

Container Name	DdsServiceEventReceiver
Parent Container	DdsConsumedServiceInstance
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents an Event receiver relying on DdsDataWriter linked to a VariableDataPrototype in a SenderReceiver interface.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataElementRef	0..1	[ECUC_Dds_00206]
DdsDataReaderRef	0..1	[ECUC_Dds_00211]

No Included Containers

⌋

[ECUC_Dds_00206] Definition of EcucForeignReferenceDef DdsDataElementRef ⌈

Parameter Name	DdsDataElementRef
Parent Container	DdsServiceEventReceiver, DdsServiceEventSender, DdsServiceFieldsFieldNotifierReceiver, DdsServiceFieldsFieldNotifierSender, DdsServiceMethodsFireAndForgetMethod
Upstream mapping defined	–
Description	This object represents a foreign reference to a VariableDataPrototype.
Multiplicity	0..1
Type	Foreign reference to VARIABLE-DATA-PROTOTYPE
Pre-Compile only	true

⌋

[ECUC_Dds_00211] Definition of EcucReferenceDef DdsDataReaderRef ⌈

Parameter Name	DdsDataReaderRef
Parent Container	DdsServiceEventReceiver, DdsServiceFieldsFieldClientGetter, DdsServiceFieldsFieldClientSetter, DdsServiceFieldsFieldNotifierReceiver, DdsServiceFieldsFieldServerGetter, DdsServiceFieldsFieldServerSetter, DdsServiceMethodsFireAndForgetMethod, DdsServiceMethodsRequestReplyMethod
Upstream mapping defined	–
Description	This object represents a reference to a DdsDataReader. It is the main object modeling the reception communication path in a Dds service instance.
Multiplicity	0..1
Type	Reference to DdsDataReader
Pre-Compile only	true

⌋

[CP_SWS_Dds_CONSTR_00951] Event receiver DdsDataElementRef ⌈If `DdsServiceEventReceiver` exists then `DdsServiceEventReceiver.DdsDataElementRef` shall be defined.⌋

[CP_SWS_Dds_CONSTR_00953] Event receiver DdsDataReaderRef ⌈if `DdsServiceEventReceiver` exists then `DdsServiceEventReceiver.DdsDataReaderRef` shall be defined.⌋

[ECUC_Dds_00222] Definition of EcucParamConfContainerDef DdsServiceFieldsFieldClient ⌈

Container Name	DdsServiceFieldsFieldClient
Parent Container	DdsConsumedServiceInstance
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle all the Fields actions on client side.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsServiceFieldsFieldClientGetter	0..1	This object collects the configuration needed to handle all the Fields Getter action on client side.
DdsServiceFieldsFieldClientSetter	0..1	This object collects the configuration needed to handle all the Fields Setter action on client side.
DdsServiceFieldsFieldNotifier Receiver	0..1	This object collects the configuration needed to handle all the Fields Notify action on client side.

⌋

[ECUC_Dds_00223] Definition of EcucParamConfContainerDef DdsServiceFields FieldNotifierReceiver ⌈

Container Name	DdsServiceFieldsFieldNotifierReceiver
Parent Container	DdsServiceFieldsFieldClient
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle all the Fields Notify action on client side.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataElementRef	0..1	[ECUC_Dds_00206]
DdsDataReaderRef	0..1	[ECUC_Dds_00211]

No Included Containers

⌋

For parameter table [ECUC_Dds_00206] DdsDataElementRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

[CP_SWS_Dds_CONSTR_00955] Field Notifier receiver DdsDataElementRef ⌈If `DdsServiceFieldsFieldNotifierReceiver` exists then `DdsServiceFields-FieldNotifierReceiver.DdsDataElementRef` shall be defined.⌋

[CP_SWS_Dds_CONSTR_00959] Field Notifier receiver DdsDataReaderRef ⌈If `DdsServiceFieldsFieldNotifierReceiver` exists then `DdsServiceFields-FieldNotifierReceiver.DdsDataReaderRef` shall be defined.⌋

[ECUC_Dds_00224] Definition of EcucParamConfContainerDef DdsServiceFields FieldClientGetter ⌈

Container Name	DdsServiceFieldsFieldClientGetter
Parent Container	DdsServiceFieldsFieldClient
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle all the Fields Getter action on client side.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DdsDataReaderRef	0..1	[ECUC_Dds_00211]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]
DdsGetOperationRef	0..1	[ECUC_Dds_00212]

No Included Containers

⌋

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

[ECUC_Dds_00212] Definition of EcucForeignReferenceDef DdsGetOperationRef ⌈

Parameter Name	DdsGetOperationRef
Parent Container	DdsServiceFieldsFieldClientGetter, DdsServiceFieldsFieldServerGetter
Upstream mapping defined	–
Description	This object represents a foreign reference to a ClientServerOperation. It models the Get operation in a Field.
Multiplicity	0..1
Type	Foreign reference to CLIENT-SERVER-OPERATION
Pre-Compile only	true

⌋

[ECUC_Dds_00225] Definition of EcucParamConfContainerDef DdsServiceFieldsFieldClientSetter ⌈

Container Name	DdsServiceFieldsFieldClientSetter
Parent Container	DdsServiceFieldsFieldClient
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This object collects the configuration needed to handle all the Fields Setter action on client side.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDataReaderRef	0..1	[ECUC_Dds_00211]
DdsDataWriterRef	0..1	[ECUC_Dds_00207]
DdsSetOperationRef	0..1	[ECUC_Dds_00213]

No Included Containers

⌋

For parameter table [ECUC_Dds_00211] DdsDataReaderRef, see definition below container DdsServiceEventReceiver.

For parameter table [ECUC_Dds_00207] DdsDataWriterRef, see definition below container DdsServiceEventSender.

[ECUC_Dds_00213] Definition of EcucForeignReferenceDef DdsSetOperationRef
⌈

Parameter Name	DdsSetOperationRef
Parent Container	DdsServiceFieldsFieldClientSetter, DdsServiceFieldsFieldServerSetter
Upstream mapping defined	–
Description	This object represents a foreign reference to a ClientServerOperation. It models the Set operation in a Field.
Multiplicity	0..1
Type	Foreign reference to CLIENT-SERVER-OPERATION
Pre-Compile only	true

⌋

10.2.3.4 DdsDomainMulticastRtpsCollection

The `DdsDomainMulticastRtpsPduCollection` container models the pool of all the user multicast lower layer PDUs (both Tx and Rx) towards lower network layers.

`Dds` employs multicast PDUs for both user traffic and discovery traffic (SPDP, refer to `DdsDomainDpMulticastRtpsPduCollection`).

Unlike unicast PDUs, which are aggregated by the generic `DdsDomainParticipant` and owned by it, multicast PDUs are not aggregated by individual `DdsDomainParticipant`. This is due to the fact that, in general, a PDU represents the unidirectional interface towards the AUTOSAR stack for `SoAdConnection`, which in this case are of multicast type. In accordance with the OMG standard [8], the configuration of a multicast connection depends exclusively on the `DdsDomainId` of the `DdsDomainParticipant` that will use it.

For the reason, the corresponding PDU must be explicitly associated with the same `DdsDomainId`.

[ECUC_Dds_00144] Definition of EcucParamConfContainerDef DdsDomainMulticastRtpsPduCollection ⌈

Container Name	DdsDomainMulticastRtpsPduCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of low-level multicast PDUs used for transporting user traffic.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DdsRtpsMulticastDataRxPdu	0..1	The multicast reception lower layer pdu used to send data from the lower network layer to DDS itself.
DdsRtpsMulticastDataTxPdu	0..1	The multicast lower layer transmission PDU used to transmit data from DDS to lower network layer.

⌋

DdsRtpsMulticastDataTxPdu

[ECUC_Dds_00151] Definition of EcucParamConfContainerDef DdsRtpsMulticastDataTxPdu ⌈

Container Name	DdsRtpsMulticastDataTxPdu	
Parent Container	DdsDomainMulticastRtpsPduCollection	
Upstream mapping defined	–	
Description	The multicast lower layer transmission PDU used to transmit data from DDS to lower network layer.	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDomainId	1	[ECUC_Dds_00138]
DdsRtpsMulticastDataTxPduId	1	[ECUC_Dds_00152]
DdsRtpsMulticastDataTxPduRef	1	[ECUC_Dds_00153]
DdsSoAdSocketConnectionRef	0..65535	[ECUC_Dds_00185]

No Included Containers

⌋

For parameter table [ECUC_Dds_00138] DdsDomainId, see definition below container DdsDomainParticipant.

For parameter table [ECUC_Dds_00152] DdsRtpsMulticastDataTxPduId, see definition below container DdsRtpsDpMulticastDataTxPdu.

For parameter table [ECUC_Dds_00153] DdsRtpsMulticastDataTxPduRef, see definition below container DdsRtpsDpMulticastDataTxPdu.

For parameter table [ECUC_Dds_00185] DdsSoAdSocketConnectionRef, see definition below container DdsRtpsDataTxPdu.

DdsRtpsMulticastDataRxPdu

[ECUC_Dds_00154] Definition of EcucParamConfContainerDef DdsRtpsMulticastDataRxPdu ⌈

Container Name	DdsRtpsMulticastDataRxPdu	
Parent Container	DdsDomainMulticastRtpsPduCollection	
Upstream mapping defined	–	
Description	The multicast reception lower layer pdu used to send data from the lower network layer to DDS itself.	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDomainId	1	[ECUC_Dds_00138]
DdsRtpsMulticastDataRxPduId	1	[ECUC_Dds_00155]
DdsRtpsMulticastDataRxPduRef	1	[ECUC_Dds_00156]

No Included Containers

⌋

For parameter table [ECUC_Dds_00138] DdsDomainId, see definition below container DdsDomainParticipant.

For parameter table [ECUC_Dds_00155] DdsRtpsMulticastDataRxPduId, see definition below container DdsRtpsDpMulticastDataRxPdu.

For parameter table [ECUC_Dds_00156] DdsRtpsMulticastDataRxPduRef, see definition below container DdsRtpsDpMulticastDataRxPdu.

10.2.3.5 DdsDomainDpMulticastRtpsPdusCollection

The `DdsDomainDpMulticastRtpsPduCollection`

[ECUC_Dds_00188] Definition of EcucParamConfContainerDef DdsDomainDpMulticastRtpsPduCollection ⌈

Container Name	DdsDomainDpMulticastRtpsPduCollection
Parent Container	DdsConfig
Upstream mapping defined	–
Description	Collection of low-level multicast PDUs used for transporting SPDP discovery traffic.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
DdsRtpsDpMulticastDataRxPdu	0..1	The multicast reception lower layer pdu used to send service discovered data from the lower network layer to DDS itself.
DdsRtpsDpMulticastDataTxPdu	0..1	The multicast lower layer transmission PDU used to transmit service discovered data from DDS to lower network layer.

⌋

DdsRtpsDpMulticastDataTxPdu

[ECUC_Dds_00189] Definition of EcucParamConfContainerDef DdsRtpsDpMulticastDataTxPdu ⌈

Container Name	DdsRtpsDpMulticastDataTxPdu	
Parent Container	DdsDomainDpMulticastRtpsPduCollection	
Upstream mapping defined	–	
Description	The multicast lower layer transmission PDU used to transmit service discovered data from DDS to lower network layer.	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DdsDomainId	1	[ECUC_Dds_00138]
DdsRtpsMulticastDataTxPduId	1	[ECUC_Dds_00152]
DdsRtpsMulticastDataTxPduRef	1	[ECUC_Dds_00153]
DdsSoAdSocketConnectionRef	0..65535	[ECUC_Dds_00185]

No Included Containers

⌋

For parameter table [ECUC_Dds_00138] DdsDomainId, see definition below container DdsDomainParticipant.

[ECUC_Dds_00152] Definition of EcucIntegerParamDef DdsRtpsMulticastDataTxPduId ⌈

Parameter Name	DdsRtpsMulticastDataTxPduId	
Parent Container	DdsRtpsDpMulticastDataTxPdu, DdsRtpsMulticastDataTxPdu	
Upstream mapping defined	–	
Description	The current pdu local id	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Dds_00153] Definition of EcucReferenceDef DdsRtpsMulticastDataTxPduRef ⌈

Parameter Name	DdsRtpsMulticastDataTxPduRef
Parent Container	DdsRtpsDpMulticastDataTxPdu, DdsRtpsMulticastDataTxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the Dds module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

For parameter table [ECUC_Dds_00185] DdsSoAdSocketConnectionRef, see definition below container DdsRtpsDataTxPdu.

DdsRtpsDpMulticastDataRxPdu

[ECUC_Dds_00190] Definition of EcucParamConfContainerDef DdsRtpsDpMulticastDataRxPdu ⌈

Container Name	DdsRtpsDpMulticastDataRxPdu	
Parent Container	DdsDomainDpMulticastRtpsPduCollection	
Upstream mapping defined	–	
Description	The multicast reception lower layer pdu used to send service discovered data from the lower network layer to DDS itself.	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDomainId	1	[ECUC_Dds_00138]
DdsRtpsMulticastDataRxPduId	1	[ECUC_Dds_00155]
DdsRtpsMulticastDataRxPduRef	1	[ECUC_Dds_00156]

No Included Containers

⌋

For parameter table [ECUC_Dds_00138] DdsDomainId, see definition below container DdsDomainParticipant.

[ECUC_Dds_00155] Definition of EcucIntegerParamDef DdsRtpsMulticastDataRxPduId ⌈

Parameter Name	DdsRtpsMulticastDataRxPduId
Parent Container	DdsRtpsDpMulticastDataRxPdu, DdsRtpsMulticastDataRxPdu
Upstream mapping defined	–
Description	The current pdu local id.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Dds_00156] Definition of EcucReferenceDef DdsRtpsMulticastDataRxPduRef ⌈

Parameter Name	DdsRtpsMulticastDataRxPduRef
Parent Container	DdsRtpsDpMulticastDataRxPdu, DdsRtpsMulticastDataRxPdu
Upstream mapping defined	–
Description	The reference to a PDU in the global PDU structure described in the AUTOSAR ECU Configuration Specification. This reference will be used by the Dds module to derive the PDU Id.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

10.2.3.6 DdsQoSPolicies

Note: This chapter is intended to describe all the possible QoS policies. Each of them can be applied only to specific entity types.

The container DdsQoSPolicies does not exist actually, there is a specific Dds<Entity_type>QoS subcontainer for each entity type that supports QoS policies. For each entity type, in the related chapter, the correct Dds<Entity_type>QoS is described.

Note: For description of specific QoS policies and the constraints that apply to them, refer to "Supported QoS" chapter of DDS OMG specification ([2]).

10.2.3.6.1 DdsUserData

[CP_SWS_Dds_01001] DDS USER_DATA semantics

Upstream requirements: FO_RS_Dds_00005

⌈If DdsUserData (the container used for DDS USER_DATA QoS policy) is configured for a specific entity (DdsDomainParticipants, DdsDataReaders or DdsDataWriters), the Dds shall provide user defined information for this entity to be exchanged with other entities in the same DDS Domain. User data information would be distributed in the context of a DomainParticipant only by means of build-in topics, not with every exchanged message.

The DdsUserDataValue buffer shall be statically configured according an external agreement between parties and it will be never modified at runtime. Being statically configured and not accessible from the application, which can not modify its content, no API is required.

Note: the configuration and meaning of USER_DATA QoS is vendor specific, it can be used to exchange any kind of information between entities. Its handling would be done in DDS middleware and it is out of scope of this document. For any details on this QoS topic, the "USER_DATA" chapter of [2] shall be taken as reference.

⌋

[ECUC_Dds_00019] Definition of EcucParamConfContainerDef DdsUserData ⌈

Container Name	DdsUserData
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsDomainParticipantQoS
Upstream mapping defined	–
Description	Describes the DDS [2] USER_DATA QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsUserDataValue	1	[ECUC_Dds_00124]

No Included Containers

⌋

[ECUC_Dds_00124] Definition of EcucStringParamDef DdsUserDataValue ⌈

Parameter Name	DdsUserDataValue
Parent Container	DdsUserData
Upstream mapping defined	–
Description	See "USER_DATA" chapter of DDS [2].
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.3.6.2 DdsTopicData

[ECUC_Dds_00106] Definition of EcucParamConfContainerDef DdsTopicData ⌈

Container Name	DdsTopicData
Parent Container	DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] TOPIC_DATA QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsTopicDataValue	1	[ECUC_Dds_00126]

No Included Containers

⌋

[ECUC_Dds_00126] Definition of EcucStringParamDef DdsTopicDataValue ⌈

Parameter Name	DdsTopicDataValue
Parent Container	DdsTopicData
Upstream mapping defined	–
Description	See "TOPIC_DATA" chapter of DDS [2].
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.3.6.3 DdsGroupData

[ECUC_Dds_00024] Definition of EcucParamConfContainerDef DdsGroupData ⌈

Container Name	DdsGroupData
Parent Container	DdsPublisherQoS, DdsSubscriberQoS
Upstream mapping defined	–
Description	Describes the DDS [2] GROUP_DATA QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsGroupDataValue	1	[ECUC_Dds_00125]

No Included Containers

⌋

[ECUC_Dds_00125] Definition of EcucStringParamDef DdsGroupDataValue ⌈

Parameter Name	DdsGroupDataValue
Parent Container	DdsGroupData
Upstream mapping defined	–
Description	See "GROUP_DATA" chapter of DDS [2].
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.3.6.4 DdsDurability

[ECUC_Dds_00034] Definition of EcucParamConfContainerDef DdsDurability ⌈

Container Name	DdsDurability
Parent Container	DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] DURABILITY QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDurabilityKind	1	[ECUC_Dds_00035]

No Included Containers

⌋

[ECUC_Dds_00035] Definition of EcucEnumerationParamDef DdsDurabilityKind ⌈

Parameter Name	DdsDurabilityKind	
Parent Container	DdsDurability	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "DURABILITY" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	PERSISTENT	See "DURABILITY" chapter of DDS[2].
	TRANSIENT	See "DURABILITY" chapter of DDS [2].
	TRANSIENT_LOCAL	See "DURABILITY" chapter of DDS [2].
	VOLATILE	See "DURABILITY" chapter of DDS [2].
Default value	VOLATILE	
Pre-Compile only	true	

⌋

10.2.3.6.5 DdsDurabilityService

[ECUC_Dds_00036] Definition of EcucParamConfContainerDef DdsDurability Service ⌈

Container Name	DdsDurabilityService
Parent Container	DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] DURABILITY_SERVICE QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDurabilityServiceCleanupDelay	1	[ECUC_Dds_00037]
DdsDurabilityServiceHistoryDepth	1	[ECUC_Dds_00119]
DdsDurabilityServiceHistoryKind	1	[ECUC_Dds_00038]
DdsDurabilityServiceMaxInstances	1	[ECUC_Dds_00121]
DdsDurabilityServiceMaxSamples	1	[ECUC_Dds_00120]
DdsDurabilityServiceMaxSamplesPerInstance	1	[ECUC_Dds_00122]

No Included Containers

⌋

[ECUC_Dds_00037] Definition of EcucFloatParamDef DdsDurabilityService CleanupDelay ⌈

Parameter Name	DdsDurabilityServiceCleanupDelay	
Parent Container	DdsDurabilityService	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "DURABILITY_SERVICE" chapter of DDS [2]. Time given in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00119] Definition of EcucIntegerParamDef DdsDurabilityServiceHistoryDepth ⌈

Parameter Name	DdsDurabilityServiceHistoryDepth	
Parent Container	DdsDurabilityService	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "DURABILITY_SERVICE" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	1	
Pre-Compile only	true	

⌋

[ECUC_Dds_00038] Definition of EcucEnumerationParamDef DdsDurabilityServiceHistoryKind ⌈

Parameter Name	DdsDurabilityServiceHistoryKind	
Parent Container	DdsDurabilityService	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "DURABILITY_SERVICE" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	KEEP_ALL	See "DURABILITY_SERVICE" chapter of DDS [2].
	KEEP_LAST	See "DURABILITY_SERVICE" chapter of DDS [2].

▽

△

Default value	KEEP_LAST
Pre-Compile only	true

⌋

[ECUC_Dds_00121] Definition of EcucIntegerParamDef DdsDurabilityService MaxInstances ⌈

Parameter Name	DdsDurabilityServiceMaxInstances
Parent Container	DdsDurabilityService
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "DURABILITY_SERVICE" chapter of DDS [2].
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65534
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00120] Definition of EcucIntegerParamDef DdsDurabilityService MaxSamples ⌈

Parameter Name	DdsDurabilityServiceMaxSamples
Parent Container	DdsDurabilityService
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "DURABILITY_SERVICE" chapter of DDS [2].
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65534
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00122] Definition of EcucIntegerParamDef DdsDurabilityService MaxSamplesPerInstance ⌈

Parameter Name	DdsDurabilityServiceMaxSamplesPerInstance
Parent Container	DdsDurabilityService
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "DURABILITY_SERVICE" chapter of DDS [2].
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65534
Default value	–
Pre-Compile only	true

⌋

10.2.3.6.6 DdsPresentation

[ECUC_Dds_00025] Definition of EcucParamConfContainerDef DdsPresentation ⌈

Container Name	DdsPresentation
Parent Container	DdsPublisherQoS, DdsSubscriberQoS
Upstream mapping defined	–
Description	Describes the DDS [2] PRESENTATION QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsPresentationAccessScope	1	[ECUC_Dds_00069]
DdsPresentationCoherentAccess	1	[ECUC_Dds_00070]
DdsPresentationOrderedAccess	1	[ECUC_Dds_00071]

No Included Containers

⌋

[ECUC_Dds_00069] Definition of EcucEnumerationParamDef DdsPresentation AccessScope ⌈

Parameter Name	DdsPresentationAccessScope	
Parent Container	DdsPresentation	
Upstream mapping defined	–	
Description	See "PRESENTATION" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	GROUP	See "PRESENTATION" chapter of DDS [2].
	INSTANCE	See "PRESENTATION" chapter of DDS [2].
	TOPIC	See "PRESENTATION" chapter of DDS [2].
Pre-Compile only	true	

⌋

[ECUC_Dds_00070] Definition of EcucBooleanParamDef DdsPresentationCoherentAccess ⌈

Parameter Name	DdsPresentationCoherentAccess
Parent Container	DdsPresentation
Upstream mapping defined	–
Description	See "PRESENTATION" chapter of DDS [2].
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	–
Pre-Compile only	true

⌋

[ECUC_Dds_00071] Definition of EcucBooleanParamDef DdsPresentationOrderedAccess ⌈

Parameter Name	DdsPresentationOrderedAccess
Parent Container	DdsPresentation
Upstream mapping defined	–
Description	See "PRESENTATION" chapter of DDS [2].
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

10.2.3.6.7 DdsDeadline

[ECUC_Dds_00039] Definition of EcucParamConfContainerDef DdsDeadline ⌈

Container Name	DdsDeadline
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] DEADLINE QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsDeadlinePeriod	1	[ECUC_Dds_00040]

No Included Containers

⌋

[ECUC_Dds_00040] Definition of EcucFloatParamDef DdsDeadlinePeriod ⌈

Parameter Name	DdsDeadlinePeriod
Parent Container	DdsDeadline
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "DEADLINE" chapter of DDS [2]. Time given in seconds.

▽

△

Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

10.2.3.6.8 DdsLatencyBudget

[ECUC_Dds_00041] Definition of EcucParamConfContainerDef DdsLatencyBudget ⌈

Container Name	DdsLatencyBudget
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] LATENCY_BUDGET QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsLatencyBudgetDuration	1	[ECUC_Dds_00042]

No Included Containers

⌋

[ECUC_Dds_00042] Definition of EcucFloatParamDef DdsLatencyBudgetDuration ⌈

Parameter Name	DdsLatencyBudgetDuration	
Parent Container	DdsLatencyBudget	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "LATENCY_BUDGET" chapter of DDS [2]. Time given in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

10.2.3.6.9 DdsOwnership

[ECUC_Dds_00043] Definition of EcucParamConfContainerDef DdsOwnership ⌈

Container Name	DdsOwnership
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] OWNERSHIP QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsOwnershipKind	1	[ECUC_Dds_00044]

No Included Containers

⌋

[ECUC_Dds_00044] Definition of EcucEnumerationParamDef DdsOwnership Kind ⌈

Parameter Name	DdsOwnershipKind	
Parent Container	DdsOwnership	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "OWNERSHIP" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	EXCLUSIVE	See "OWNERSHIP" chapter of DDS [2].
	SHARED	See "OWNERSHIP" chapter of DDS [2].
Default value	SHARED	
Pre-Compile only	true	

⌋

10.2.3.6.10 DdsOwnershipStrength

[ECUC_Dds_00045] Definition of EcucParamConfContainerDef DdsOwnership Strength ⌈

Container Name	DdsOwnershipStrength
Parent Container	DdsDataWriterQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] OWNERSHIP_STRENGTH QoS policy.
Multiplicity	0..1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsOwnershipStrengthValue	1	[ECUC_Dds_00046]

No Included Containers

⌋

[ECUC_Dds_00046] Definition of EcucIntegerParamDef DdsOwnershipStrength Value ⌈

Parameter Name	DdsOwnershipStrengthValue	
Parent Container	DdsOwnershipStrength	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "OWNERSHIP_STRENGTH" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	0	
Pre-Compile only	true	

⌋

10.2.3.6.11 DdsLiveliness

[ECUC_Dds_00047] Definition of EcucParamConfContainerDef DdsLiveliness ⌈

Container Name	DdsLiveliness
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] LIVELINESS QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsLivelinessLeaseDuration	1	[ECUC_Dds_00049]
DdsLivenessKind	1	[ECUC_Dds_00048]

No Included Containers

⌋

[ECUC_Dds_00049] Definition of EcucFloatParamDef DdsLivelinessLeaseDuration ⌈

Parameter Name	DdsLivelinessLeaseDuration	
Parent Container	DdsLiveliness	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "LIVELINESS" chapter of DDS [2]. Time given in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dds_00048] Definition of EcucEnumerationParamDef DdsLivenessKind ⌈

Parameter Name	DdsLivenessKind	
Parent Container	DdsLiveliness	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "LIVELINESS" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	AUTOMATIC	See "LIVELINESS" chapter of DDS [2].
	MANUAL_BY_PARTICIPANT	See "LIVELINESS" chapter of DDS [2].
	MANUAL_BY_TOPIC	See "LIVELINESS" chapter of DDS [2].
Default value	AUTOMATIC	
Pre-Compile only	true	

⌋

10.2.3.6.12 DdsTimeBasedFilter

[ECUC_Dds_00087] Definition of EcucParamConfContainerDef DdsTimeBasedFilter ⌈

Container Name	DdsTimeBasedFilter
Parent Container	DdsDataReaderQoS
Upstream mapping defined	–
Description	Describes the DDS [2] TIME_BASED_FILTER QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsTimeBasedFilterMinimumSeparation	1	[ECUC_Dds_00088]

No Included Containers

⌋

[ECUC_Dds_00088] Definition of EcucFloatParamDef DdsTimeBasedFilterMinimumSeparation ⌈

Parameter Name	DdsTimeBasedFilterMinimumSeparation	
Parent Container	DdsTimeBasedFilter	
Upstream mapping defined	–	
Description	See "TIME_BASED_FILTER" chapter of DDS [2]. Time given in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

10.2.3.6.13 DdsPartition

[ECUC_Dds_00026] Definition of EcucParamConfContainerDef DdsPartition ⌈

Container Name	DdsPartition
Parent Container	DdsPublisherQoS, DdsSubscriberQoS
Upstream mapping defined	–
Description	Describes the DDS [2] PARTITION QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DdsPartitionName	1..65534	[ECUC_Dds_00072]

No Included Containers

⌋

[ECUC_Dds_00072] Definition of EcucStringParamDef DdsPartitionName ⌈

Parameter Name	DdsPartitionName
Parent Container	DdsPartition
Upstream mapping defined	–
Description	See "PARTITION" chapter of DDS [2].

▽

△

Multiplicity	1..65534
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.3.6.14 DdsReliability

[ECUC_Dds_00050] Definition of EcucParamConfContainerDef DdsReliability ⌈

Container Name	DdsReliability
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] RELIABILITY QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsReliabilityKind	1	[ECUC_Dds_00051]
DdsReliabilityMaxBlockingTime	1	[ECUC_Dds_00052]

No Included Containers

⌋

[ECUC_Dds_00051] Definition of EcucEnumerationParamDef DdsReliabilityKind ⌈

Parameter Name	DdsReliabilityKind	
Parent Container	DdsReliability	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "RELIABILITY" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BEST_EFFORT	See "RELIABILITY" chapter of DDS [2].
	RELIABLE	See "RELIABILITY" chapter of DDS [2].
Pre-Compile only	true	

⌋

[ECUC_Dds_00052] Definition of EcucFloatParamDef DdsReliabilityMaxBlockingTime ⌈

Parameter Name	DdsReliabilityMaxBlockingTime
Parent Container	DdsReliability
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "RELIABILITY" chapter of DDS [2]. Time given in seconds.
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. 65.534]
Default value	0.1
Pre-Compile only	true

⌋

10.2.3.6.15 DdsTransportPriority

[ECUC_Dds_00053] Definition of EcucParamConfContainerDef DdsTransportPriority ⌈

Container Name	DdsTransportPriority
Parent Container	DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] TRANSPORT_PRIORITY QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsTransportPriorityValue	1	[ECUC_Dds_00054]

No Included Containers

⌋

[ECUC_Dds_00054] Definition of EcucIntegerParamDef DdsTransportPriorityValue ⌈

Parameter Name	DdsTransportPriorityValue
Parent Container	DdsTransportPriority
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "TRANSPORT_PRIORITY" chapter of DDS [2].
Multiplicity	1
Type	EcucIntegerParamDef

▽

△

Range	0 .. 65534	
Default value	0	
Pre-Compile only	true	

⌋

10.2.3.6.16 DdsLifespan

[ECUC_Dds_00055] Definition of EcucParamConfContainerDef DdsLifespan ⌈

Container Name	DdsLifespan
Parent Container	DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] LIFESPAN QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsLifespanDuration	1	[ECUC_Dds_00056]

No Included Containers

⌋

[ECUC_Dds_00056] Definition of EcucFloatParamDef DdsLifespanDuration ⌈

Parameter Name	DdsLifespanDuration	
Parent Container	DdsLifespan	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "LIFESPAN" chapter of DDS [2]. Time given in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.534]	
Default value	–	
Pre-Compile only	true	

⌋

10.2.3.6.17 DdsDestinationOrder

[ECUC_Dds_00057] Definition of EcucParamConfContainerDef DdsDestination Order ⌈

Container Name	DdsDestinationOrder
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] DESTINATION_ORDER QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DdsDestinationOrderKind	1	[ECUC_Dds_00058]

No Included Containers

⌋

[ECUC_Dds_00058] Definition of EcucEnumerationParamDef DdsDestinationOrderKind ⌈

Parameter Name	DdsDestinationOrderKind	
Parent Container	DdsDestinationOrder	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "DESTINATION_ORDER" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BY_RECEPTION_TIMESTAMP	See "DESTINATION_ORDER" chapter of DDS [2].
	BY_SOURCE_TIMESTAMP	See "DESTINATION_ORDER" chapter of DDS [2].
Default value	BY_RECEPTION_TIMESTAMP	
Pre-Compile only	true	

⌋

10.2.3.6.18 DdsHistory

[ECUC_Dds_00059] Definition of EcucParamConfContainerDef DdsHistory ⌈

Container Name	DdsHistory
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] HISTORY QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsHistoryKind	1	[ECUC_Dds_00060]
DdsHistoryOrderDepth	1	[ECUC_Dds_00063]

No Included Containers

⌋

[ECUC_Dds_00060] Definition of EcucEnumerationParamDef DdsHistoryKind ⌈

Parameter Name	DdsHistoryKind	
Parent Container	DdsHistory	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "HISTORY" chapter of DDS [2].	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	KEEP_ALL	See "HISTORY" chapter of DDS [2].
	KEEP_LAST	See "HISTORY" chapter of DDS [2].
Default value	KEEP_LAST	
Pre-Compile only	true	

⌋

[ECUC_Dds_00063] Definition of EcucIntegerParamDef DdsHistoryOrderDepth ⌈

Parameter Name	DdsHistoryOrderDepth	
Parent Container	DdsHistory	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "HISTORY" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	1	
Pre-Compile only	true	

⌋

10.2.3.6.19 DdsResourceLimits

[ECUC_Dds_00061] Definition of EcucParamConfContainerDef DdsResourceLimits ⌈

Container Name	DdsResourceLimits
Parent Container	DdsDataReaderQoS, DdsDataWriterQoS, DdsTopicQoS
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Describes the DDS [2] RESOURCE_LIMITS QoS policy.

▽

△

Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsResouceLimitsMaxInstances	1	[ECUC_Dds_00064]
DdsResouceLimitsMaxSamples	1	[ECUC_Dds_00062]
DdsResouceLimitsMaxSamplesPerInstance	1	[ECUC_Dds_00065]

No Included Containers

⌋

[ECUC_Dds_00064] Definition of EcucIntegerParamDef DdsResouceLimitsMaxInstances ⌈

Parameter Name	DdsResouceLimitsMaxInstances	
Parent Container	DdsResourceLimits	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "RESOURCE_LIMITS" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	65534	
Pre-Compile only	true	

⌋

[ECUC_Dds_00062] Definition of EcucIntegerParamDef DdsResouceLimitsMaxSamples ⌈

Parameter Name	DdsResouceLimitsMaxSamples	
Parent Container	DdsResourceLimits	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	See "RESOURCE_LIMITS" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	65534	
Pre-Compile only	true	

⌋

[ECUC_Dds_00065] Definition of EcucIntegerParamDef DdsResouceLimitsMax SamplesPerInstance ⌈

Parameter Name	DdsResouceLimitsMaxSamplesPerInstance
Parent Container	DdsResourceLimits
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	See "RESOURCE_LIMITS" chapter of DDS [2].
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65534
Default value	65534
Pre-Compile only	true

⌋

10.2.3.6.20 DdsEntityFactory

[ECUC_Dds_00027] Definition of EcucParamConfContainerDef DdsEntityFactory ⌈

Container Name	DdsEntityFactory
Parent Container	DdsDomainParticipantQoS, DdsPublisherQoS, DdsSubscriberQoS
Upstream mapping defined	–
Description	Describes the DDS [2] ENTITY_FACTORY QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsEntityFactoryAutoenableCreatedEntities	1	[ECUC_Dds_00073]

No Included Containers

⌋

[ECUC_Dds_00073] Definition of EcucBooleanParamDef DdsEntityFactoryAutoenableCreatedEntities ⌈

Parameter Name	DdsEntityFactoryAutoenableCreatedEntities
Parent Container	DdsEntityFactory
Upstream mapping defined	–
Description	See "ENTITY_FACTORY" chapter of DDS [2].
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true

▽

△

Pre-Compile only	true

⌋

10.2.3.6.21 DdsWriterDataLifecycle

[ECUC_Dds_00066] Definition of EcucParamConfContainerDef DdsWriterDataLifecycle ⌈

Container Name	DdsWriterDataLifecycle
Parent Container	DdsDataWriterQoS
Upstream mapping defined	–
Description	Describes the DDS [2] WRITER_DATA_LIFECYCLE QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DdsAutodisposeUnregisteredInstances	1	[ECUC_Dds_00067]

No Included Containers

⌋

[ECUC_Dds_00067] Definition of EcucBooleanParamDef DdsAutodisposeUnregisteredInstances ⌈

Parameter Name	DdsAutodisposeUnregisteredInstances
Parent Container	DdsWriterDataLifecycle
Upstream mapping defined	–
Description	See "WRITER_DATA_LIFECYCLE" chapter of DDS [2].
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

10.2.3.6.22 DdsReaderDataLifecycle

[ECUC_Dds_00095] Definition of EcucParamConfContainerDef DdsReaderDataLifecycle ⌈

Container Name	DdsReaderDataLifecycle
Parent Container	DdsDataReaderQoS
Upstream mapping defined	–
Description	Describes the DDS [2] READER_DATA_LIFECYCLE QoS policy.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DdsAutopurgeDisposedSamplesDelay	1	[ECUC_Dds_00097]
DdsAutopurgeNowriterSamplesDelay	1	[ECUC_Dds_00096]

No Included Containers

⌋

[ECUC_Dds_00097] Definition of EcucIntegerParamDef DdsAutopurgeDisposedSamplesDelay ⌈

Parameter Name	DdsAutopurgeDisposedSamplesDelay	
Parent Container	DdsReaderDataLifecycle	
Upstream mapping defined	–	
Description	See "READER_DATA_LIFECYCLE" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	65534	
Pre-Compile only	true	

⌋

[ECUC_Dds_00096] Definition of EcucIntegerParamDef DdsAutopurgeNowriterSamplesDelay ⌈

Parameter Name	DdsAutopurgeNowriterSamplesDelay	
Parent Container	DdsReaderDataLifecycle	
Upstream mapping defined	–	
Description	See "READER_DATA_LIFECYCLE" chapter of DDS [2].	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65534	
Default value	65534	
Pre-Compile only	true	

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral.

A Not applicable requirements

[SWS_Dds_NA_00000] Not applicable requirements

Upstream requirements: SRS_BSW_00004, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00336, SRS_BSW_00345, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00386, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00399, SRS_BSW_00403, SRS_BSW_00406, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00425, SRS_BSW_00452, SRS_BSW_00458, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00490, SRS_BSW_00491

⌈This item lists all the not applicable requirements for this specification.⌋

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R25-11

B.1.1 Added Specification Items in R25-11

Number	Heading
[CP_SWS_Dds_00720]	DdsAppDataTxPdu Meta Data handling
[CP_SWS_Dds_00721]	DdsRtpsDataRxPdu Meta Data handling
[CP_SWS_Dds_00722]	DdsRtpsDataTxPdu Meta Data handling in DdsDataSender context
[CP_SWS_Dds_00723]	DdsRtpsDataTxPdu Meta Data handling in unidirectional service elements context
[CP_SWS_Dds_00737]	RTPS transmission PDUs for discovery purpose
[CP_SWS_Dds_00738]	RTPS reception PDUs for discovery purpose
[CP_SWS_Dds_00900]	Configuration and manipulation of DdsTopic referenced by a DdsDataWriter referenced by a DdsServiceEventSender object
[CP_SWS_Dds_00901]	Configuration and manipulation of DdsTopic referenced by a DdsDataWriter in a Field Notifier context on server side
[CP_SWS_Dds_00902]	Configuration and manipulation of DdsTopic referenced by a DdsData Reader referenced by a DdsServiceEventReceiver object
[CP_SWS_Dds_00903]	Configuration and manipulation of DdsTopic referenced by a DdsData Reader in a Field Notifier context on client side
[CP_SWS_Dds_00904]	Configuration and manipulation of DdsTopics referenced in a Request and Response Method context on client side
[CP_SWS_Dds_00905]	Configuration and manipulation of DdsTopics referenced in a Request and Response Method context on server side
[CP_SWS_Dds_00906]	Handling of "Fire and Forget" methods with arguments
[CP_SWS_Dds_00907]	Handling of "Fire and Forget" methods without arguments
[CP_SWS_Dds_00908]	Configuration and manipulation of DdsTopics referenced by a DdsDataWriter or by a DdsDataReader, in a Fields Getter/Setter context on client side

△

Number	Heading
[CP_SWS_Dds_00909]	Configuration and manipulation of DdsTopics referenced by a DdsDataWriter or by a DdsDataReader, in a Fields Getter/Setter context on server side
[CP_SWS_Dds_00921]	DdsAppDataRxPdu format in bidirectional service interface elements context
[CP_SWS_Dds_00925]	DdsAppDataTxPdu requestId mapping on server side in bidirectional service interface elements context
[CP_SWS_Dds_00926]	DdsAppDataTxPdu requestId mapping on client side in bidirectional service interface elements context
[CP_SWS_Dds_00927]	DdsAppDataRxPdu requestId mapping on server side in bidirectional service interface elements context
[CP_SWS_Dds_00928]	DdsAppDataRxPdu requestId mapping on client side in bidirectional service interface elements context
[CP_SWS_Dds_00930]	returnValue mapping on server side in bidirectional service elements context
[CP_SWS_Dds_00931]	returnValue mapping on client side in bidirectional service elements context
[CP_SWS_Dds_01010]	SPDP support
[CP_SWS_Dds_01011]	SEDP support
[CP_SWS_Dds_01012]	Dynamic Service Instance Discovery support
[CP_SWS_Dds_01013]	Dds actions after a remote Domain Participant discover event
[CP_SWS_Dds_01014]	Dds actions after a remote Domain Participant removal event
[CP_SWS_Dds_01015]	Discovery behaviour on network fail
[CP_SWS_Dds_01019]	Setting remote Domain Participant address
[CP_SWS_Dds_01020]	Resetting remote Domain Participant address
[CP_SWS_Dds_01031]	Remote participant connection access
[ECUC_Dds_00183]	Definition of EcucParamConfContainerDef DdsDomainParticipantDpUnicastRtpsPduCollection
[ECUC_Dds_00184]	Definition of EcucParamConfContainerDef DdsRtpsDpDataTxPdu
[ECUC_Dds_00185]	Definition of EcucReferenceDef DdsSoAdSocketConnectionRef
[ECUC_Dds_00187]	Definition of EcucParamConfContainerDef DdsRtpsDpDataRxPdu
[ECUC_Dds_00188]	Definition of EcucParamConfContainerDef DdsDomainDpMulticastRtpsPduCollection
[ECUC_Dds_00189]	Definition of EcucParamConfContainerDef DdsRtpsDpMulticastDataTxPdu
[ECUC_Dds_00190]	Definition of EcucParamConfContainerDef DdsRtpsDpMulticastDataRxPdu

▽

△

Number	Heading
[ECUC_Dds_00191]	Definition of EcucReferenceDef DdsRxQueueDpUnicastPduRef
[ECUC_Dds_00192]	Definition of EcucReferenceDef DdsRxQueueDpMulticastPduRef
[ECUC_Dds_00193]	Definition of EcucParamConfContainerDef DdsProvidedServiceInstance
[ECUC_Dds_00194]	Definition of EcucStringParamDef DdsServiceInterfaceId
[ECUC_Dds_00195]	Definition of EcucIntegerParamDef DdsServiceInstanceId
[ECUC_Dds_00196]	Definition of EcucEnumerationParamDef DdsResourceIdentifierType
[ECUC_Dds_00197]	Definition of EcucEnumerationParamDef DdsDiscoveryType
[ECUC_Dds_00198]	Definition of EcucStringParamDef DdsFieldRequestTopicName
[ECUC_Dds_00199]	Definition of EcucStringParamDef DdsFieldReplyTopicName
[ECUC_Dds_00200]	Definition of EcucStringParamDef DdsMethodRequestTopicName
[ECUC_Dds_00201]	Definition of EcucStringParamDef DdsMethodReplyTopicName
[ECUC_Dds_00202]	Definition of EcucStringParamDef DdsTransportProtocol
[ECUC_Dds_00203]	Definition of EcucIntegerParamDef DdsServiceMajorVersion
[ECUC_Dds_00204]	Definition of EcucIntegerParamDef DdsServiceMinorVersion
[ECUC_Dds_00205]	Definition of EcucParamConfContainerDef DdsServiceEventSender
[ECUC_Dds_00206]	Definition of EcucForeignReferenceDef DdsDataElementRef
[ECUC_Dds_00207]	Definition of EcucReferenceDef DdsDataWriterRef
[ECUC_Dds_00208]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldServer
[ECUC_Dds_00209]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldNotifier Sender
[ECUC_Dds_00210]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldServer Getter
[ECUC_Dds_00211]	Definition of EcucReferenceDef DdsDataReaderRef
[ECUC_Dds_00212]	Definition of EcucForeignReferenceDef DdsGetOperationRef
[ECUC_Dds_00213]	Definition of EcucForeignReferenceDef DdsSetOperationRef
[ECUC_Dds_00214]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldServer Setter
[ECUC_Dds_00215]	Definition of EcucParamConfContainerDef DdsServiceMethodsRequest ReplyMethod
[ECUC_Dds_00216]	Definition of EcucForeignReferenceDef DdsOperationRef
[ECUC_Dds_00217]	Definition of EcucParamConfContainerDef DdsServiceMethodsFireAnd ForgetMethod
[ECUC_Dds_00218]	Definition of EcucForeignReferenceDef DdsTriggerRef
[ECUC_Dds_00219]	Definition of EcucParamConfContainerDef DdsConsumedServiceInstance
[ECUC_Dds_00220]	Definition of EcucIntegerParamDef DdsRequiredServiceInstanceId
[ECUC_Dds_00221]	Definition of EcucParamConfContainerDef DdsServiceEventReceiver
[ECUC_Dds_00222]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldClient
[ECUC_Dds_00223]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldNotifier Receiver

▽

△

Number	Heading
[ECUC_Dds_00224]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldClient Getter
[ECUC_Dds_00225]	Definition of EcucParamConfContainerDef DdsServiceFieldsFieldClient Setter
[ECUC_Dds_00228]	Definition of EcucFloatParamDef DdsParticipantLeaseDuration
[ECUC_Dds_00229]	Definition of EcucFloatParamDef DdsSpdpResendPeriod

Table B.1: Added Specification Items in R25-11

B.1.2 Changed Specification Items in R25-11

Number	Heading
[CP_SWS_Dds_00852]	Dds_Transmit - Error conditions
[ECUC_Dds_00005]	Definition of EcucParamConfContainerDef DdsConfig
[ECUC_Dds_00012]	Definition of EcucParamConfContainerDef DdsDomainParticipant

Table B.2: Changed Specification Items in R25-11

B.1.3 Deleted Specification Items in R25-11

none

B.1.4 Added Constraints in R25-11

Number	Heading
[CP_SWS_Dds_CONSTR_00950]	Event sender DdsDataElementRef
[CP_SWS_Dds_CONSTR_00951]	Event receiver DdsDataElementRef
[CP_SWS_Dds_CONSTR_00952]	Event sender DdsDataWriterRef

▽

△

Number	Heading
[CP_SWS_Dds_CONSTR_00953]	Event receiver DdsDataReaderRef
[CP_SWS_Dds_CONSTR_00954]	Field Notifier sender DdsDataElementRef
[CP_SWS_Dds_CONSTR_00955]	Field Notifier receiver DdsDataElementRef
[CP_SWS_Dds_CONSTR_00956]	Fire and Forget Method DdsDataElementRef
[CP_SWS_Dds_CONSTR_00957]	Fire and Forget Method DdsTriggerRef
[CP_SWS_Dds_CONSTR_00958]	Field Notifier sender DdsDataWriterRef
[CP_SWS_Dds_CONSTR_00959]	Field Notifier receiver DdsDataReaderRef
[CP_SWS_Dds_CONSTR_00960]	DdsDataWriter employed in a service interface element
[CP_SWS_Dds_CONSTR_00961]	DdsDataReader employed in a service interface element
[CP_SWS_Dds_CONSTR_00970]	DdsDomainParticipant can aggregate only one DdsProvidedServiceInstance
[CP_SWS_Dds_CONSTR_00971]	DdsDomainParticipant can aggregate only one DdsConsumedServiceInstance

Table B.3: Added Constraints in R25-11

B.1.5 Changed Constraints in R25-11

none

B.1.6 Deleted Constraints in R25-11

Number	Heading
[CP_SWS_ Dds_ CONSTR_ 00725]	No data serialization

Table B.4: Deleted Constraints in R25-11

B.2 Traceable item history of this document according to AUTOSAR Release R26-11

B.2.1 Added Specification Items in R26-11

Number	Heading
[ECUC_Dds_00230]	Definition of EcucBooleanParamDef DdsImmediateTxConfirmation
[SWS_Dds_00844]	Definition of API function Dds_ReleaseRxBuffer
[SWS_Dds_10001]	Application PDU immediate transmission confirmation to the upper layer
[SWS_Dds_10002]	Asynchronous copying of Tx application I-PDU to local buffer
[SWS_Dds_10003]	Synchronous copying of Tx application I-PDU to local buffer
[SWS_Dds_10004]	Tx application I-PDU processing after successful asynchronous copy to local buffer
[SWS_Dds_10005]	Copying of Tx application I-PDU (`KeepLocalPduBuffer` set to `TRUE`) to local buffer fails
[SWS_Dds_10006]	Asynchronous copying of Rx RTPS I-PDU to local buffer
[SWS_Dds_10007]	Synchronous copying of Rx RTPS I-PDU to local buffer
[SWS_Dds_10008]	Rx RTPS I-PDU processing after successful asynchronous copy to local buffer
[SWS_Dds_10009]	Rx RTPS I-PDU buffer release after local buffer copy
[SWS_Dds_10010]	Rx RTPS I-PDU buffer copy failure
[SWS_Dds_10011]	RTPS I-PDU memory alignment for transmission

Table B.5: Added Specification Items in R26-11

B.2.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_Dds_00132]	Definition of EcucParamConfContainerDef DdsAppDataTxPdu

Table B.6: Changed Specification Items in R26-11

B.2.3 Deleted Specification Items in R26-11

Number	Heading
[CP_SWS_Dds_00820]	Definition of API function Dds_GetVersionInfo
[CP_SWS_Dds_00821]	Dds_GetVersion - Null versioninfo pointer

Table B.7: Deleted Specification Items in R26-11

B.2.4 Added Constraints in R26-11

none

B.2.5 Changed Constraints in R26-11

none

B.2.6 Deleted Constraints in R26-11

none

C Referenced Meta Classes

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

Class	**ApplicationError**			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ClientServerInterface.possibleError			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).

Table C.1: ApplicationError

Class	**ArgumentDataPrototype**			
Note	An argument of an operation, carries direction and implementation information.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ClientServerOperation.argument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirectionEnum	0..1	attr	This attribute specifies the direction of the argument.
serverArgumentImplPolicy	ServerArgumentImplPolicyEnum	0..1	attr	This defines how the argument type of the servers `RunnableEntity` is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value `useArgumentType` for primitive arguments and structures.

Table C.2: ArgumentDataPrototype

Class	**ClientServerInterface**			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operation	ClientServerOperation	*	aggr	`ClientServerOperation`(s) of this `ClientServerInterface`. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=operation, operation.shortName, operation.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	**ClientServerInterface**			
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface. **Tags:** atp.Splitkey=possibleError, possibleError.shortName

Table C.3: ClientServerInterface

Class	**ClientServerOperation**			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.command, *AtpClassifier*.atpFeature, ClientServerInterface.operation, DiagnosticDataElementInterface.read, DiagnosticDataIdentifierInterface.read, DiagnosticDataIdentifierInterface.write, DiagnosticExtendedDataRecordInterface.provide, DiagnosticRoutineInterface.requestResult, DiagnosticRoutineInterface.start, DiagnosticRoutineInterface.stop, ServiceInterface.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this `ClientServerOperation`. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
diagArgIntegrity	Boolean	0..1	attr	This attribute shall only be used in the implementation of diagnostic routines to support the case where input and output arguments are allocated in a shared buffer and might unintentionally overwrite input arguments by tentative write operations to output arguments. This situation can happen during sliced execution or while output parameters are arrays (call by reference). The value true means that the `ClientServerOperation` is aware of the usage of a shared buffer and takes precautions to avoid unintentional overwrite of input arguments. If the attribute does not exist or is set to false the `ClientServerOperation` does not have to consider the usage of a shared buffer. This Attribute is only used by the AUTOSAR Classic Platform.
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation. **Tags:** atp.Splitkey=possibleError This Attribute is only used by the AUTOSAR Classic Platform.

Table C.4: ClientServerOperation

Class	**Collection**
Note	This meta-class specifies a collection of elements. A collection can be utilized to express additional aspects for a set of elements. Note that Collection is an ARElement. Therefore it is applicable e.g. for EvaluatedVariant, even if this is not obvious. Usually the category of a Collection is "SET". On the other hand, a Collection can also express an arbitrary relationship between elements. This is denoted by the category "RELATION" (see also [TPS_GST_00347]). In this case the collection represents an association from "sourceElement" to "targetElement" in the role "role". **Tags:** atp.recommendedPackage=Collections

▽

△

Class	**Collection**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
autoCollect	AutoCollectEnum	0..1	attr	This attribute reflects how far the referenced objects are part of the collection. **Tags:** xml.sequenceOffset=20
collected Instance	AtpFeature	*	iref	This instance ref supports the use case that a particular instance is part of the collection. **Tags:** atp.Splitkey=collectedInstance, collectedInstance.context Element, collectedInstance.target xml.sequenceOffset=60 **InstanceRef implemented by:** AnyInstanceRef
collection Semantics	NameToken	0..1	attr	Provides the ability to express the semantics of a Collection depending on the intended use case. The collectionSemantics is specified as a NameToken which must be agreed by all stakeholders. **Tags:** xml.sequenceOffset=25
element	Identifiable	*	ref	This is an element in the collection. Note that Collection itself is collectable. Therefore collections can be nested. In case of category="RELATION" this represents the target end of the relation. **Tags:** atp.Splitkey=element xml.sequenceOffset=40
elementRole	Identifier	0..1	attr	This attribute allows to denote a particular role of the collection. Note that the applicable semantics shall be mutually agreed between the two parties. In particular it denotes the role of element in the context of sourceElement. **Tags:** xml.sequenceOffset=30
sourceElement	Identifiable	*	ref	Only if Category = “RELATION”. This represents the source of a relation. **Tags:** atp.Splitkey=sourceElement xml.sequenceOffset=50
sourceInstance	AtpFeature	*	iref	Only if Category = “RELATION”. This represents the source instance of a relation. **Tags:** atp.Splitkey=sourceInstance, sourceInstance.context Element, sourceInstance.target xml.sequenceOffset=70 **InstanceRef implemented by:** AnyInstanceRef

Table C.5: Collection

Class	**DdsCpConsumedServiceInstance**
Note	This meta-class represents the ability to describe the existence and configuration of a consumed (required) service instance in a concrete implementation on top of DDS. **Tags:** atp.Status=candidate
Base	*ARObject*, *AbstractServiceInstance*, *DdsCpServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**DdsCpConsumedServiceInstance**			
Aggregated by	ServiceInstanceCollectionSet.serviceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
consumedEvent	DdsServiceInstance EventCp	*	aggr	Collection of consumed events. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=consumedEvent, consumedEvent.short Name, consumedEvent.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
consumedField	DdsServiceInstance FieldCp	*	aggr	Collection of consumed fields. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=consumedField, consumedField.shortName, consumedField.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
consumed Operation	DdsServiceInstance OperationCp	*	aggr	Collection of consumed operations. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=consumedOperation, consumed Operation.shortName, consumedOperation.variation Point.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
localUnicast Address	ApplicationEndpoint	0..1	ref	The local address over which the Service is consumed. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=localUnicastAddress, localUnicast Address.applicationEndpoint, localUnicast Address.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime xml.namePlural=LOCAL-UNICAST-ADDRESSES
requiredService InstanceId	PositiveInteger	0..1	attr	This attribute describes the required service instance ID. **Tags:** atp.Status=candidate
staticRemote Multicast Address	ApplicationEndpoint	0..1	ref	This reference defines the remote multicast address of the Service provider. This reference shall ONLY be used if the remote multicast address of the server is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=staticRemoteMulticastAddress, staticRemote MulticastAddress.applicationEndpoint, staticRemote MulticastAddress.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime xml.name Plural=STATIC-REMOTE-MULTICAST-ADDRESSES

▽

△

Class	**DdsCpConsumedServiceInstance**			
staticRemote UnicastAddress	ApplicationEndpoint	0..1	ref	This reference defines the remote unicast address of the Service provider. This reference shall ONLY be used if the remote unicast address of the server is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=staticRemoteUnicastAddress, staticRemote UnicastAddress.applicationEndpoint, staticRemote UnicastAddress.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime xml.name Plural=STATIC-REMOTE-UNICAST-ADDRESSES

Table C.6: DdsCpConsumedServiceInstance

Class	**DdsCpProvidedServiceInstance**			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of DDS. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AbstractServiceInstance*, *DdsCpServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceCollectionSet.serviceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localUnicast Address	ApplicationEndpoint	0..1	ref	The local address over which the Service is provided. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=localUnicastAddress, localUnicast Address.applicationEndpoint, localUnicast Address.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime xml.namePlural=LOCAL-UNICAST-ADDRESSES
providedEvent	DdsServiceInstance EventCp	*	aggr	Collection of provided events. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=providedEvent, providedEvent.shortName, providedEvent.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
providedField	DdsServiceInstance FieldCp	*	aggr	Collection of provided fields. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=providedField, providedField.shortName, providedField.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
provided Operation	DdsServiceInstance OperationCp	*	aggr	Collection of provided operations. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=providedOperation, providedOperation.short Name, providedOperation.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild

▽

△

Class	**DdsCpProvidedServiceInstance**			
providedService InstanceId	PositiveInteger	0..1	attr	This attribute describes the provided service instance ID. **Tags:** atp.Status=candidate
staticRemote Multicast Address	ApplicationEndpoint	0..1	ref	This reference defines the remote multicast address of Service consumers. This reference shall ONLY be used if the remote multicast address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=staticRemoteMulticastAddress, staticRemote MulticastAddress.applicationEndpoint, staticRemote MulticastAddress.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime xml.name Plural=STATIC-REMOTE-MULTICAST-ADDRESSES
staticRemote UnicastAddress	ApplicationEndpoint	*	ref	This reference defines the remote unicast addresses of Service consumers. This reference shall ONLY be used if the remote unicast address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=staticRemoteUnicastAddress, staticRemote UnicastAddress.applicationEndpoint, staticRemote UnicastAddress.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime xml.name Plural=STATIC-REMOTE-UNICAST-ADDRESSES

Table C.7: DdsCpProvidedServiceInstance

Class	**ISignalIPdu**			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iPduTimingSpecification, iPduTiming Specification.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**ISignalIPdu**			
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalToPduMapping, iSignalToPdu Mapping.shortName, iSignalToPduMapping.variation Point.shortLabel vh.latestBindingTime=postBuild
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table C.8: ISignalIPdu

Class	**ImplementationDataType**			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Autosar DataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to true, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of `ImplementationDataTypeElement` is subject to variability with the purpose to support the conditional existence of elements inside a `ImplementationDataType` representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Tags:** atp.Splitkey=symbolProps, symbolProps.short Name
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table C.9: ImplementationDataType

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*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ClientServerInterface, *DataInterface*, ModeSwitchInterface, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isService	Boolean	0..1	attr	This flag is set if the `PortInterface` is to be used for communication between an • `ApplicationSwComponentType` or • `ServiceProxySwComponentType` or • `SensorActuatorSwComponentType` or • `ComplexDeviceDriverSwComponentType` • `ServiceSwComponentType` • `EcuAbstractionSwComponentType` and a `ServiceSwComponentType` (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime This Attribute is only used by the AUTOSAR Classic Platform.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service. This Attribute is only used by the AUTOSAR Classic Platform.

Table C.10: PortInterface

Class	**SenderReceiverInterface**			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	*	aggr	The data elements of this `SenderReceiverInterface`. **Tags:** atp.Splitkey=dataElement, dataElement.shortName
invalidationPolicy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement **Tags:** atp.Splitkey=invalidationPolicy
metaDataItemSet	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with `dataElement`s of the enclosing `SenderReceiverInterface` **Tags:** atp.Splitkey=metaDataItemSet

Table C.11: SenderReceiverInterface

Class	**SystemSignal**			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. **Tags:** atp.recommendedPackage=SystemSignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicLength	Boolean	0..1	attr	The length of dynamic length signals is variable in run-time. Only a maximum length of such a signal is specified in the configuration (attribute length in ISignal element).
physicalProps	SwDataDefProps	0..1	aggr	Specification of the physical representation. **Tags:** atp.Splitkey=physicalProps

Table C.12: SystemSignal

Class	**TransformationTechnology**			
Note	A TransformationTechnology is a transformer inside a transformer chain. **Tags:** xml.namePlural=TRANSFORMATION-TECHNOLOGIES			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DataTransformationSet.transformationTechnology			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bufferProperties	BufferProperties	0..1	aggr	Aggregation of the mandatory BufferProperties. **Tags:** atp.Splitkey=bufferProperties
hasInternal State	Boolean	0..1	attr	This attribute defines whether the Transformer has an internal state or not.
needsOriginal Data	Boolean	0..1	attr	Specifies whether this transformer gets access to the SWC's original data.
protocol	String	0..1	attr	Specifies the protocol that is implemented by this transformer.
transformation Description	Transformation Description	0..1	aggr	A transformer can be configured with transformer specific parameters which are represented by the Transformer Description. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=transformationDescription, transformation Description.variationPoint.shortLabel vh.latestBindingTime=postBuild
transformer Class	TransformerClassEnum	0..1	attr	Specifies to which transformer class this transformer belongs.
version	String	0..1	attr	Version of the implemented protocol.

Table C.13: TransformationTechnology

Class	**Trigger**
Note	A trigger which is provided (i.e. released) or required (i.e. used to activate something) in the given context.
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.releasedTrigger, BswModuleDescription.required Trigger, ServiceInterface.trigger, TriggerInterface.trigger

▽

△

Class	Trigger			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swImplPolicy	SwImplPolicyEnum	0..1	attr	This attribute, when set to value queued, allows for a queued processing of Triggers. This Attribute is only used by the AUTOSAR Classic Platform.
triggerPeriod	MultidimensionalTime	0..1	aggr	Optional definition of a period in case of a periodically (time or angle) driven external trigger. **Tags:** atp.Splitkey=triggerPeriod This Attribute is only used by the AUTOSAR Classic Platform.

Table C.14: Trigger

Class	TriggerInterface			
Note	A trigger interface declares a number of triggers that can be sent by an trigger source. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
trigger	Trigger	*	aggr	The Trigger of this trigger interface. **Tags:** atp.Splitkey=trigger, trigger.shortName

Table C.15: TriggerInterface

Class	VariableDataPrototype			
Note	A `VariableDataPrototype` represents a formalized generic piece of information that is typically mutable by the application software layer. `VariableDataPrototype` is used in various contexts and the specific context gives the otherwise generic `VariableDataPrototype` a dedicated semantics.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.indication, *AtpClassifier*.atpFeature, BswInternalBehavior.arTypedPerInstanceMemory, BswModuleDescription.providedData, BswModuleDescription.requiredData, BulkNvDataDescriptor.bulkNvBlock, DiagnosticSovdAccessArgument.contentObject, *InternalBehavior*.staticMemory, NvBlockDescriptor.ramBlock, NvDataInterface.nvData, SenderReceiverInterface.dataElement, ServiceInterface.event, SwcInternalBehavior.arTypedPerInstanceMemory, SwcInternalBehavior.explicitInterRunnableVariable, SwcInternalBehavior.implicitInterRunnableVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype **Tags:** atp.Splitkey=initValue

Table C.16: VariableDataPrototype

D Detailed description of Service Interface elements

D.1 Events

Figure D.1 summarises how **Event**s are handled in the AUTOSAR Classic Layered Architecture, following the Service Interface to `PortInterface` mappings prescribed by section 10.1.1 of [13].

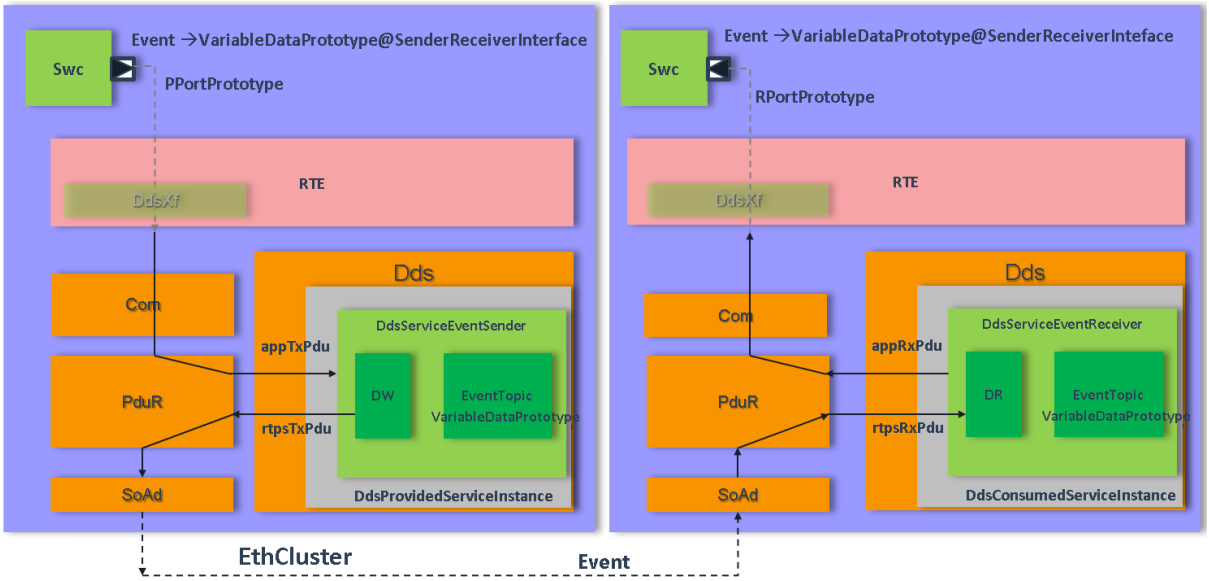


Figure D.1: Event architectural implementation with Dds

In this case, `Dds` could even perform a cast from raw data using `DdsTopicImplementationType`. This is the same behaviour used for data oriented communication path.

Note. The `DdsXf` box being "greyed out" in the picture refers to the fact that the module may or may not exist. Basically the only use case where the `DdsXf` could not be used is the one using `Com` with a `SenderReceiverInterface` with a `VariableDataPrototype` typed with a byte array.

D.2 Methods

This section summarises how **Method**s are handled in the AUTOSAR Classic Layered Architecture.

For detail also refer to 10.1.2 [13].

D.2.1 Request and Response Methods

In the Dds Module, Service Interface request and response **Method**s are implemented following the DDS-RPC standard described in [14].

Figure D.2 shows the architecture.

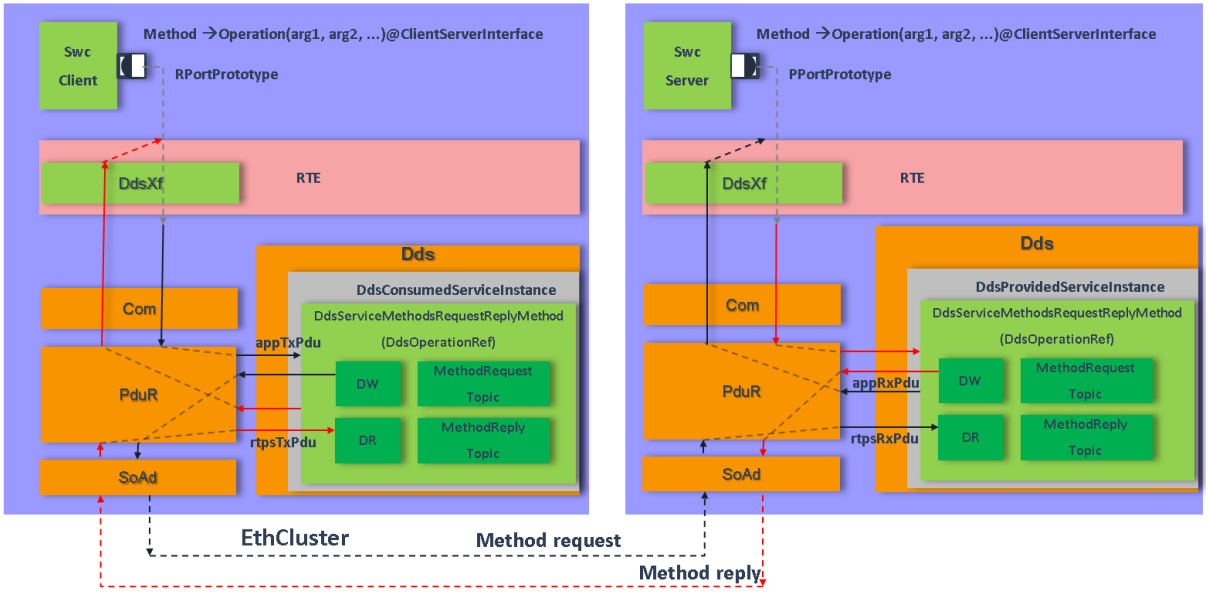


Figure D.2: Method architectural implementation with Dds

D.2.2 Fire and Forget Method with Arguments

The main difference between **Fire and Forget Method with arguments** and **Method** mentioned in Chapter D.2.1 is that the first ones don't expect a response from server.

Figure D.3 shows the architecture.

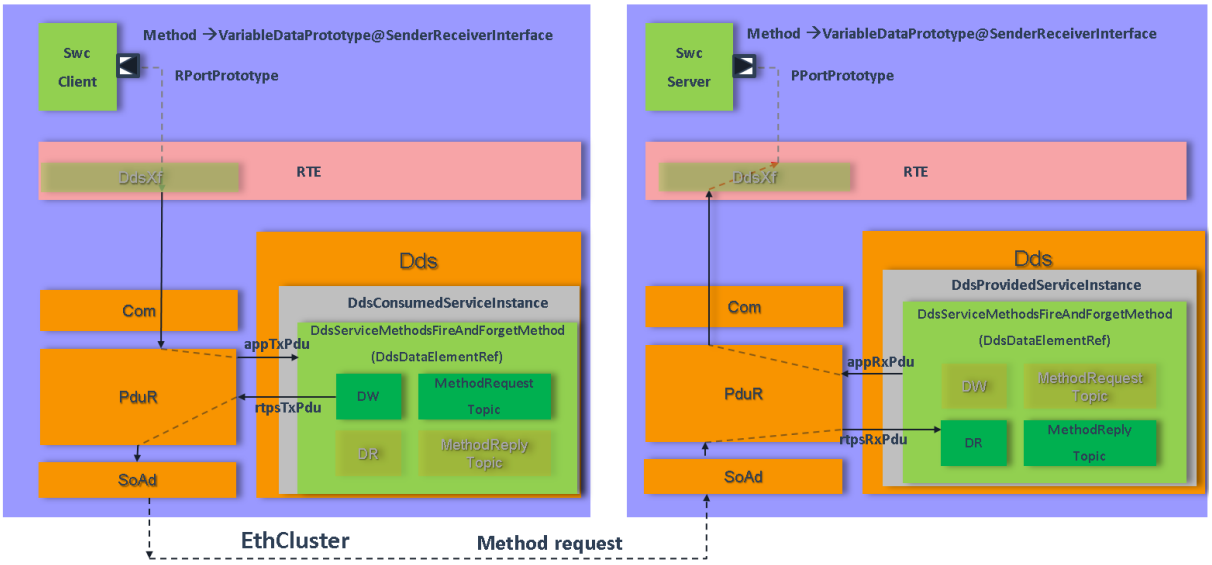


Figure D.3: Fire and Forget Method with arguments architectural implementation with Dds

D.2.3 Fire and Forget Method without arguments

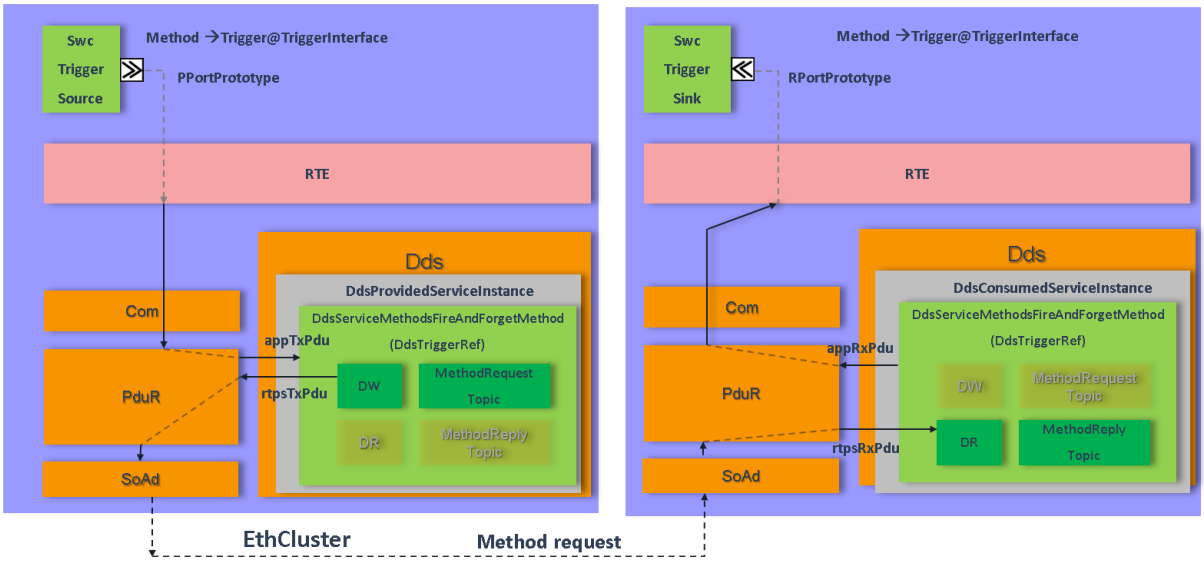


Figure D.4: Fire and Forget Method without arguments architectural implementation with Dds

Fire and Forget Method without arguments is a special case of Fire and Forget Method described at Chapter D.2.2. In AUTOSAR classic platform this kind of behaviour is implemented by means of a `TriggerInterface`. A client SW-C, acting as `Trigger` source, sends a service request to a server SW-C acting as `Trigger` sink.

D.3 Fields

From the functional point of view, **Field**s mix **Event**s and **Method**s communication paradigms. Figure D.5, Figure D.6, Figure D.7 show how **Field**s are handled in the AUTOSAR Classic Layered Architecture.

The following figures describe respectively: **Field Notifier**, **Field Getter** and **Field Setter**.

For detail refer to 10.1.4 [13].

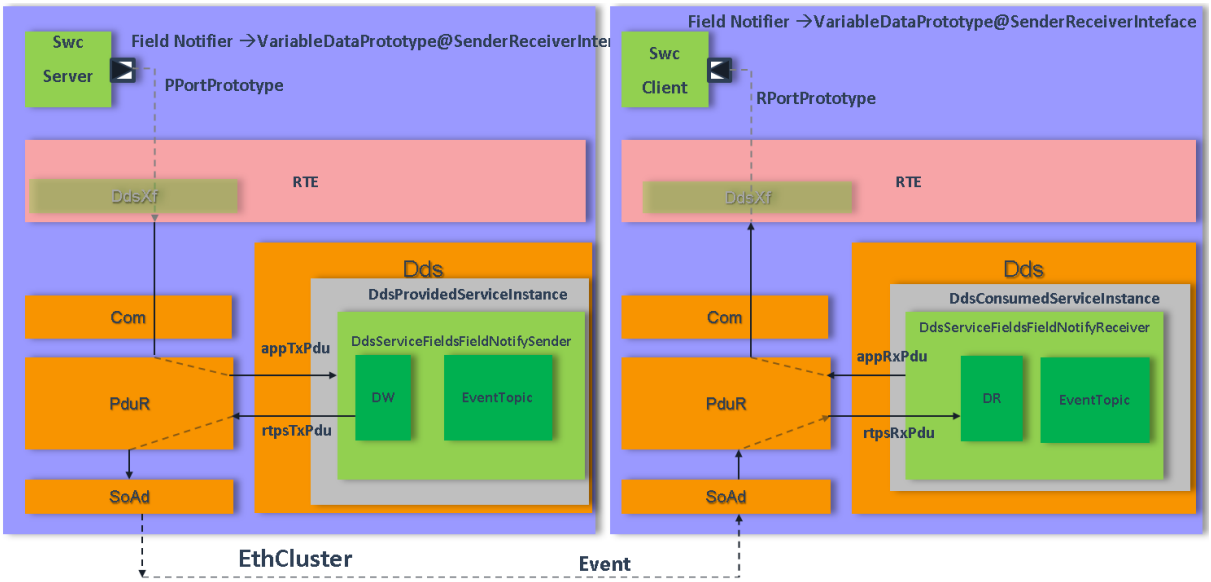


Figure D.5: Field Notifier architectural implementation with Dds

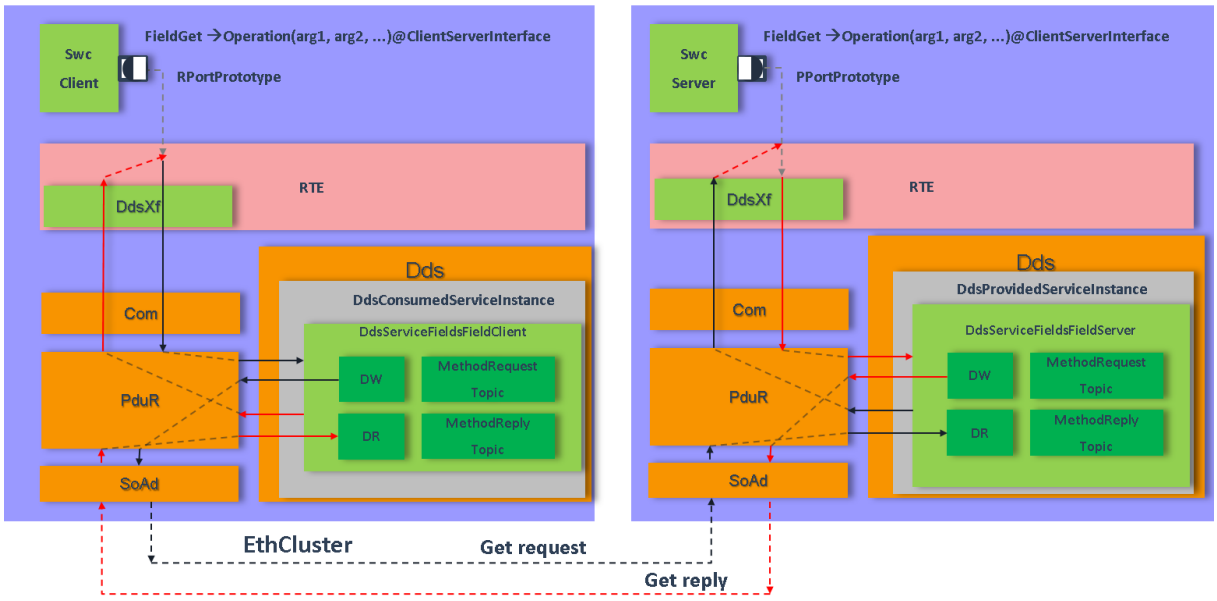


Figure D.6: Field Getter architectural implementation with Dds

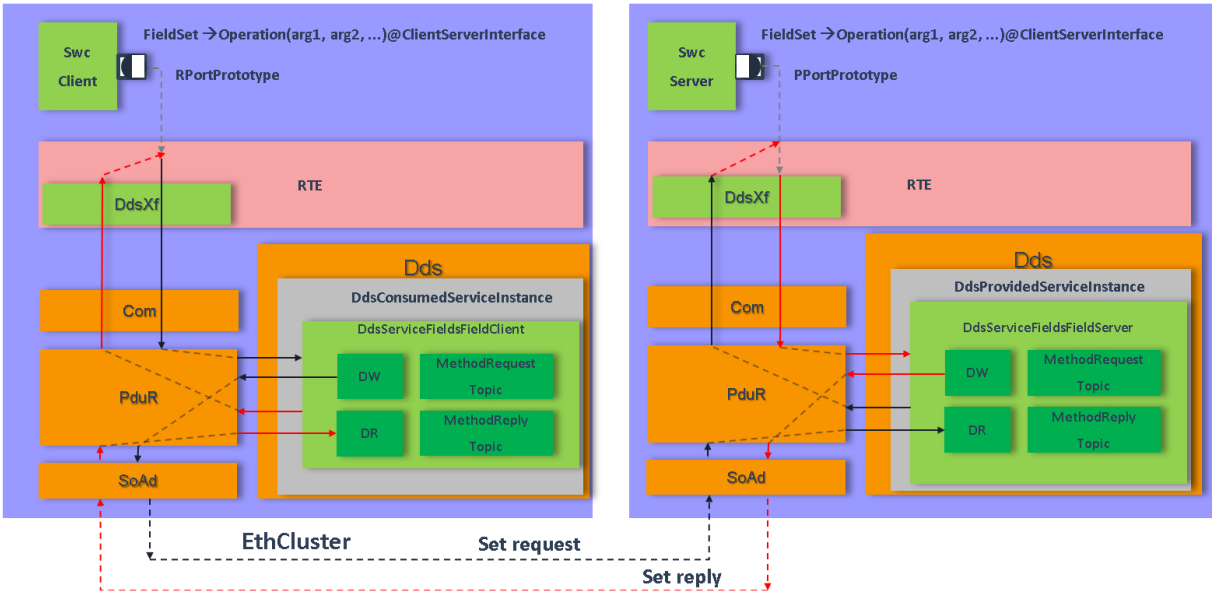


Figure D.7: Field Setter architectural implementation with Dds

E Bidirectional service interface element functional communication scheme

In order to interface the DDS middleware with the AUTOSAR CP `ClientServerInterface` communication style, `Dds` needs to map the `requestId` and the `returnValue` into DDS middleware resources and vice-versa.

Regarding the `requestId`, on the client side, `Dds` expects a `requestId` in the `DdsAppDataTxPdu`, which must be mapped into the outgoing `DdsRtpsDataTxPdu`; it must then perform the inverse mapping from the response `DdsRtpsDataRxPdu` and extract the `requestId` necessary for the `RTE` to identify the correct service instance.

Likewise, on the server side, `Dds` must extract the `requestId` from the received request `DdsRtpsDataRxPdu` and map it into the outgoing `DdsAppDataRxPdu`.

Figure E.1 shows the example of mapping on a client side for Request and Response Method.

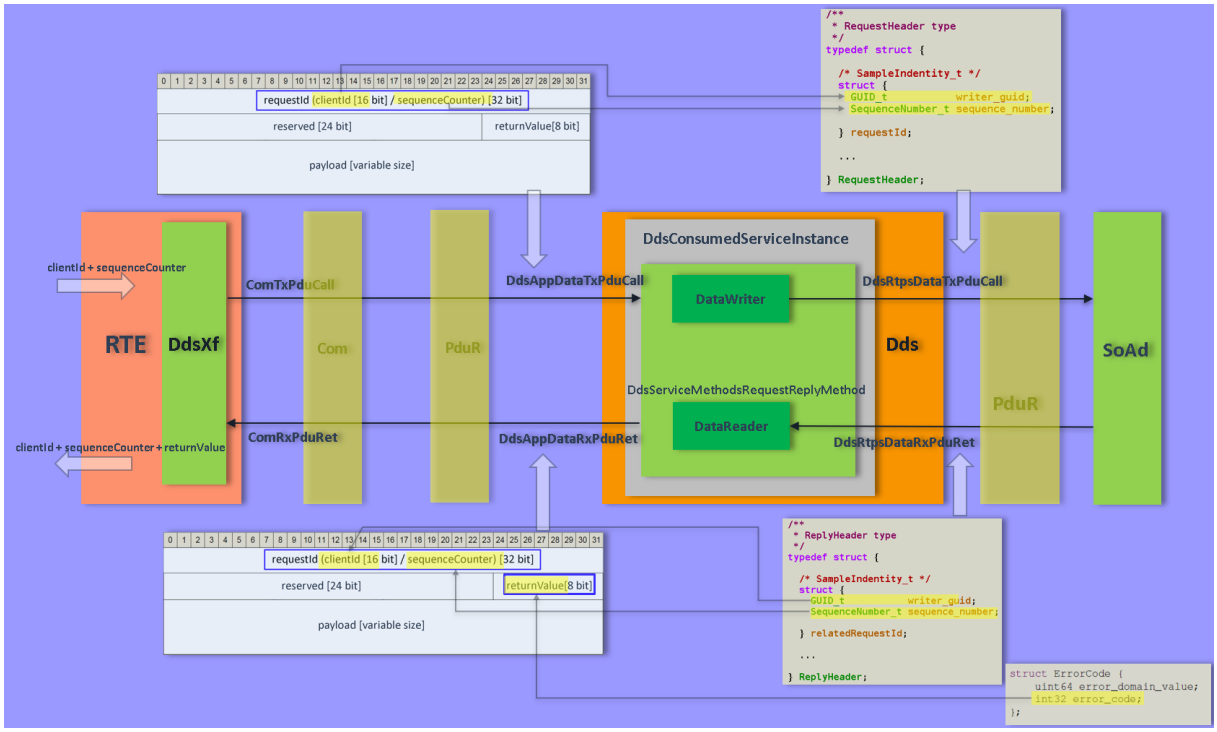


Figure E.1: Mapping the requestId to the internal DDS middleware resources on client side

Note. The `requestId` is expected to be filled with the `TransactionHandle` generated by the RTE. It allows a client to differentiate multiple calls to the same method. Therefore, the `requestId` has to be unique for a single client and server combination only. When generating a response message, the server has to copy the `requestId` from the request to the response message. This allows the client to map a response to the issued request even with more than one request outstanding.

The `returnValue` is expected to be filled with the `ApplicationError` of the executed `ClientServerOperation` on server side.

F Examples

F.1 Transport layer configuration

The transport layer and more generally the entire configuration required to ensure connectivity with Dds, via its interface composed of RTPS PDUs, constitutes a critical element, regardless of whether the scenario is entirely static or dynamic. Once the SoAd is able to provide all necessary connections for each remote Domain Participant, it is essential that Dds precisely define the relationship between the DdsRtpsDataTxPdu and the destination remote participant.

The following examples illustrate some possible configurations.

In the first example, the assumptions below must be made:

- in a certain DdsDomainParticipant only one DdsRtpsDataTxPdu is connected to a SoAdPduRoute as a pass-trough via PduR
- the SoAdPduRoute defined at previous point is the sole owned by the SoAdConnectionGroup used by the DdsDomainParticipant
- the SoAdPduRoute aggregates a number of SoAdPduRouteDest major or equal to the maximum number of remote Domain Participants that can be handled.
- the DdsDataSender and the **unidirectional service interface elements** shall use MetaData according to [CP_SWS_Dds_00722], [CP_SWS_Dds_00723]

Figure F.1 shows a typical example of SoAd configuration using unicast user traffic and multicast discovery traffic, according to the previous rules.

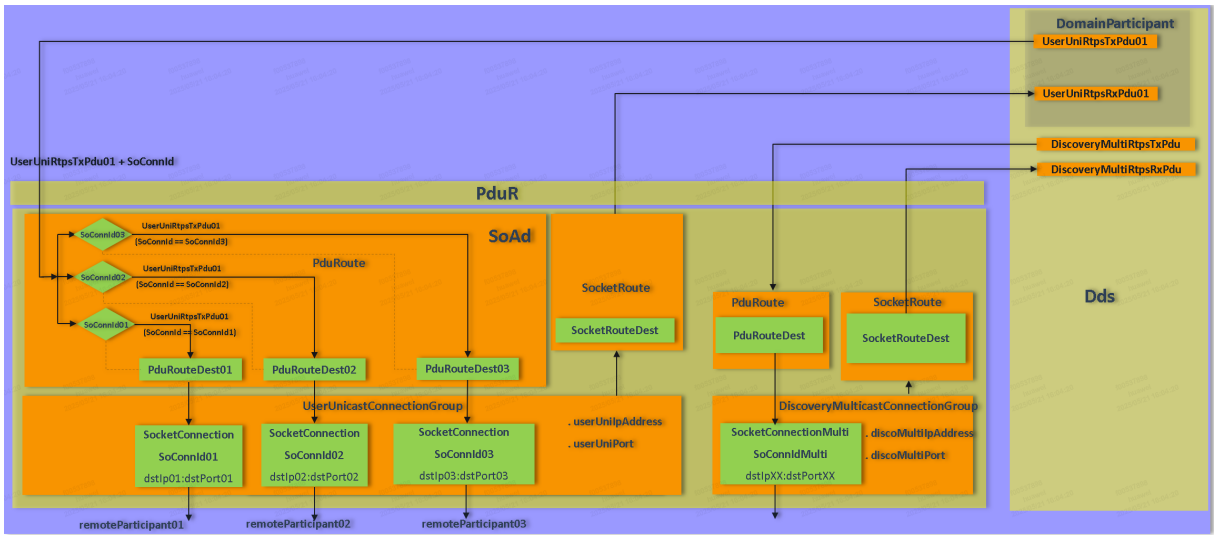


Figure F.1: An example of SoAd configuration with only one DdsRtpsDataTxPdu covering all the SoAdConnections

The Figure F.2 shows different example using more than one DdsRtpsDataTxPdus. DdsRtpsDataTxPdu01 is able to handle only SoConnId01 and SoConnId02, while DdsRtpsDataTxPdu02 can handle SoConnId03, so that the Dds should know which is the proper DdsRtpsDataTxPdu to use to reach the required remote Domain Participant.

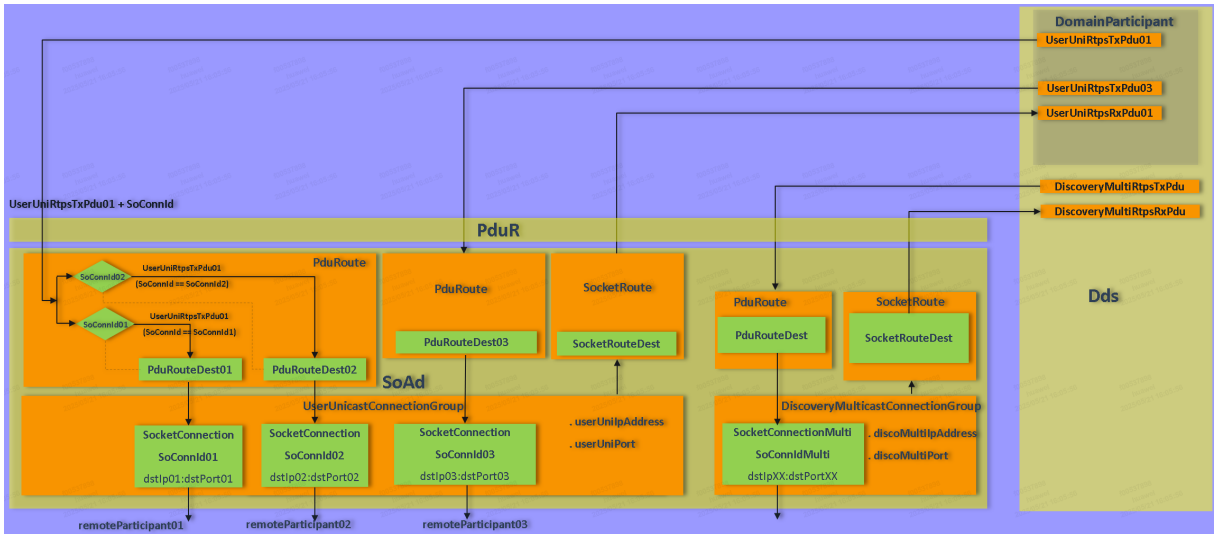


Figure F.2: An example of SoAd configuration with a couple of DdsRtpsDataTxPdus

The most general approach to the problem is resolved by establishing a direct link between the generic DdsRtpsDataTxPdu and the SoAdConnection employed for transmission. In this way, Dds can access and, if necessary, modify the configuration of that SoAdConnection.

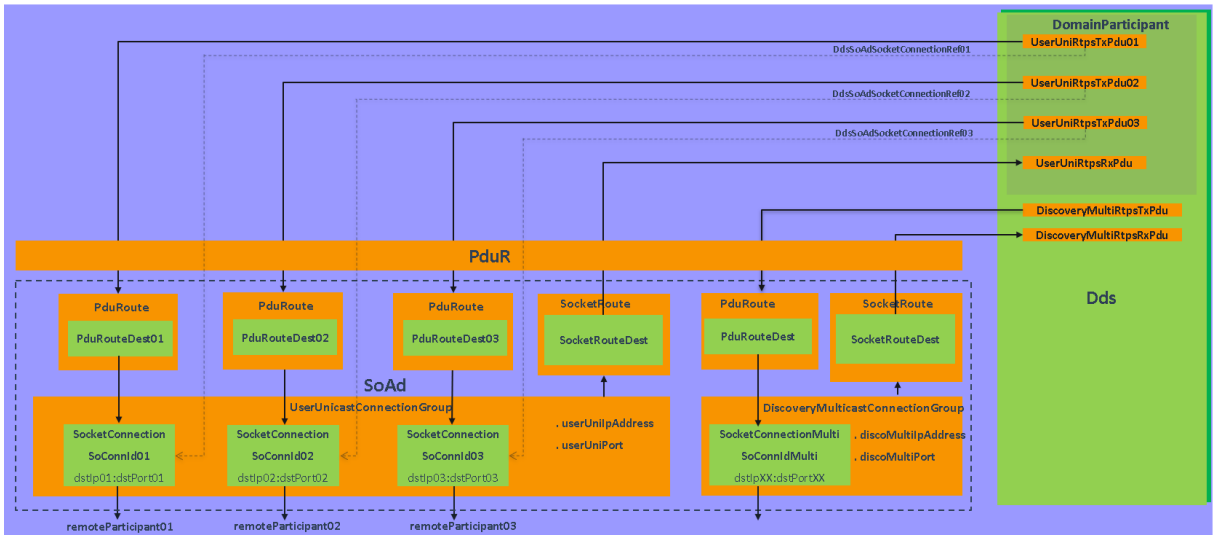


Figure F.3: An example of SoAd configuration with a couple of DdsRtpsDataTxPdus using direct reference

F.1.1 Discovery connections configuration

This chapter illustrates a possible configuration of the discovery connections. It also provides a brief mention to the typical behavior of the full discovery procedure, in which user traffic and SEDP traffic are handled over unicast connections, while SPDP traffic is transmitted via the default DDS multicast group [8].

The initial phase of the discovery process is handled by the Simple Participant Discovery Protocol (SPDP), which is responsible for detecting all active nodes associated with the specified `domainId`. Communication initialization is performed via a statically preconfigured multicast interface, identified by `SocketConnectionMultiSoConnIdMulti`, utilizing the transport channel defined by the PDU pair: `SedpMultiRtpsRxPdu`, `SedpMultiRtpsTxPdu`.

A generic local `DdsDomainParticipant` transmits an announcement message related to a specific `domainId` to the multicast group configured for SPDP (`239.255.0.1`). This message includes a list of IP addresses and ports (known as locators) that remote participants can use to establish communication with the originating participant (the list of locators in our example would contain the two pairs `192.168.0.1:7411` for user traffic and `192.168.0.1:7410` needed for the following SEDP procedure).

Upon receiving the discovery information, the local participant evaluates compatibility with the remote participant. A match is considered valid if both share the same `DdsTopic` and at least one compatible partition. If no match is found, the discovery data is discarded, and the remote participant is not considered active in the local context.

Once two participants have discovered each other, the SEDP (Simple Endpoint Discovery Protocol) phase begins, enabling mutual identification of `DataWriters` and `DataReaders`. During this phase, the unicast connections required for SEDP UDP traffic are established.

When a remote `DataWriter` or `DataReader` is detected, the DDS participant checks whether the local application contains a corresponding entity. A match between local and remote entities is valid only if: the `DataReader` and `DataWriter` share the same `DdsTopic`, they use the same data type and they have compatible `DdsDataWriterQoS`s, `DdsDataReaderQoS`s.

Communication between a local `DdsDataWriter`/`DdsDataReader` and the remote `DataReader`/`DataWriter` can only occur if both applications have linked their respective local entities to the matching remote entities. In other words, a bilateral agreement on the connection is required. At the end of this phase, the unicast connections related to user traffic are established.

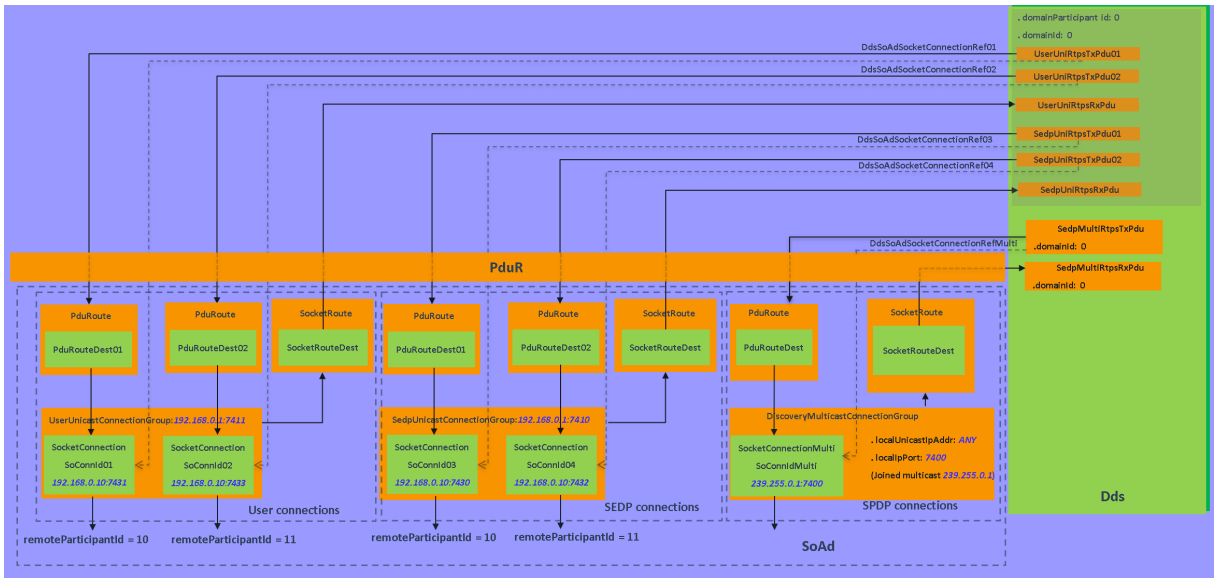


Figure F.4: An example of SoAd configuration of SPDP and SEDP connections

Discovery connections								
Traffic role	Traffic type	Port	Connection					
			Local UDP port		Local endpoint		Remote endpoint	
					Local domainId participantId	Local IP address port	Remote domainId participantId	Remote IP address port
User	Unicast	PB + DG*domainId + d3 + PG*participantId	7400 + 250*(0) + 11 + 2*(0)	7411	0:0	192.168.0.1:7411	0:10	192.168.0.10:7431
							0:11	192.168.0.11:7433
Discovery	Unicast (SEDP)	PB + DG*domainId + d1 + PG*participantId	7400 + 250*(0) + 10 + 2*(0)	7410	0:0	192.168.0.1:7410	0:10	192.168.0.10:7430
							0:11	192.168.0.11:7432
	Multicast (SPDP)	PB + DG*domainId + d0	7400 + 250*(0)	7400	0	xxx.xxx.x.x:7400	0	239.255.0.1:7400
NOTE (refer to OMG DDSI-RTPS 2.5)	The port number expressions use the following parameters: • DG = DomainId Gain • PG = ParticipantId Gain • PB = Port Base number • d0, d1, d2, d3 = additional offsets	. Implementations must expose these parameters so they can be customized by the user. . In order to enable out-of-the-box interoperability, the following default values must be used: PB = 7400; DG = 250; PG = 2; d0 = 0; d1 = 10; d2 = 1; d3 = 11. . Given UDP port numbers are limited to 64K, the above defaults enable the use of about 230 domains with up to 120 Participants per node per domain. . *domainId* is the identifier of the domainId object . *domainParticipantId* is the identifier of DDS participant ECU object						

Figure F.5: Discovery connections

F.2 SOME/IP DDS mixed architecture

SOME/IP and DDS protocols can coexist providing two different approaches to service-oriented communication.

Figure F.6 shows the AUTOSAR Classic platform Inter-ECU communication stack architecture relying on SOME/IP and `Dds` modules from the point of view of the server SW-C.

A single `ClientServerInterface` mapping should manage two couples of `SystemSignal`, one for each `TransformationTechnology`. So that, a single SW-C is able to handle service requests coming from SOME/IP and DDS clients.

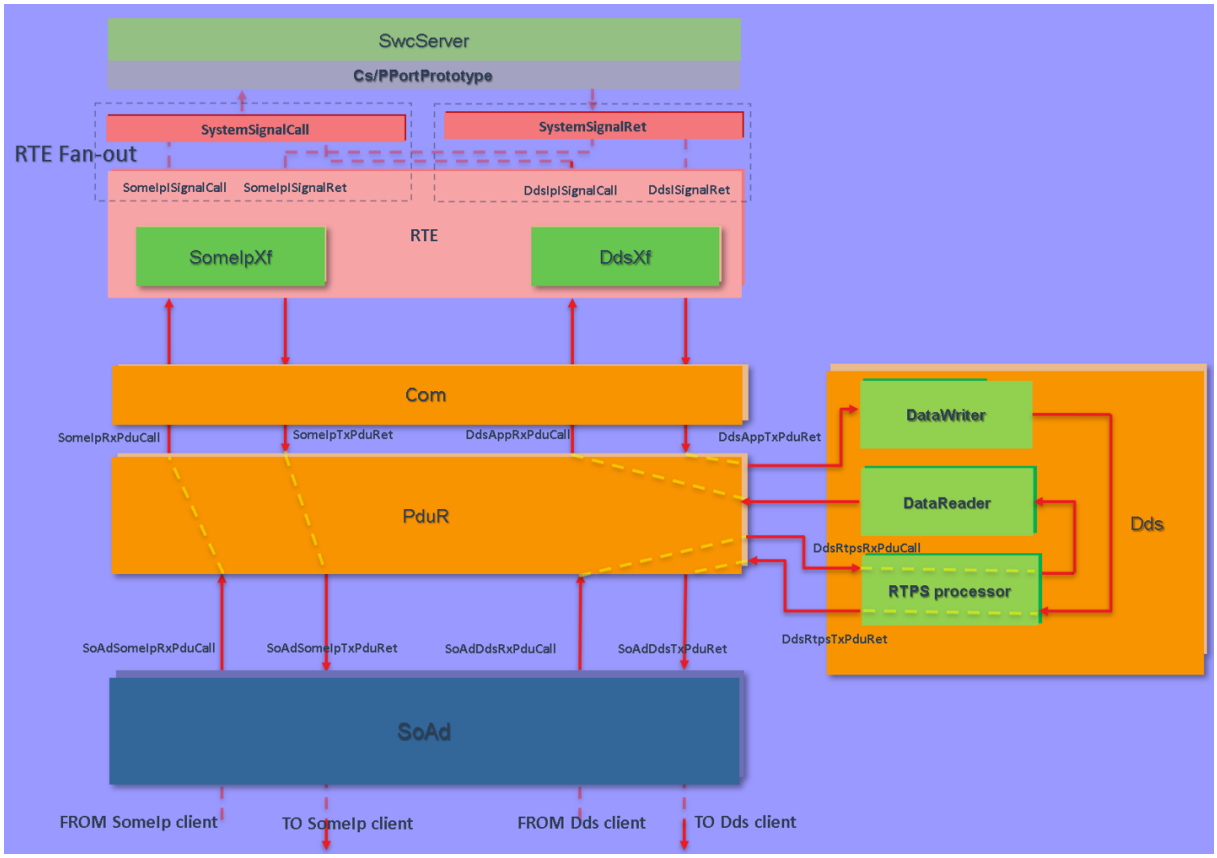


Figure F.6: Server side: architecture using SOME/IP and DDS

The case of two **Method** calls is shown in Figure F.7.

The communication path of a SOME/IP client request is highlighted in white and the one from a DDS client in yellow. They are received on different pairs of `<IP Address, port>` and the path selection is done statically with PDUs. It is up to the RTE to choose the proper path for the response.

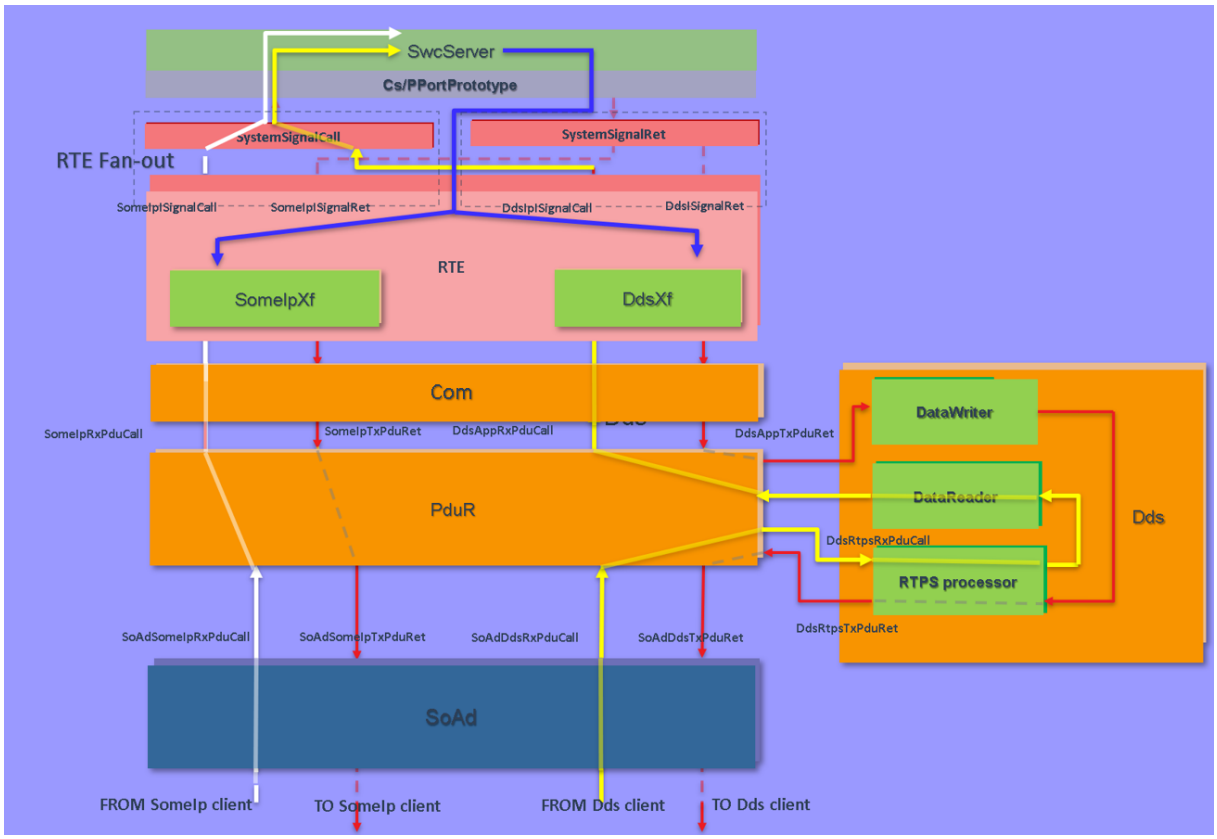


Figure F.7: Server side: response path selection