Document Title	Specification of Software Cluster Connection module
Document Category	Software specification (SWS)
Document UID	974
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	• Set requirement statuses to valid • Clarify implementation aspects of Proxy Modules • Corrections and editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Join Com and LdCom proxy to common Com Proxy • Clarify implementation aspects of Proxy Modules • Corrections and editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Remove NvM_SetBlockLockStatus API support • Clenaup EcuC spec item IDs • Corrections and editorial changes of existing features
2023-11-23	R23-11	AUTOSAR Release Management	• Support SW Cluster individual configuration ID check in NVM proxy • Requirements of R20-11 functionality are set to valid • Corrections and editorial changes of existing features
2022-11-24	R22-11	AUTOSAR Release Management	• Corrections and editorial changes of existing features



△

2021-11-25	R21-11	AUTOSAR Release Management	added support for • signal and SOME/IP based communication from Software Components located in an Application Software Cluster • connection to diagnostic services as Dem, FiM, Dcm. • Software Cluster Service Resources for connection of WdgM instances in different Software Clusters
2020-11-30	R20-11	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 19

2 Acronyms and abbreviations 20

3 Related documentation 26
3.1 Input documents & related standards and norms . . . 26
3.2 Related specification . . . 27

4 Constraints and assumptions 28
4.1 Limitations . . . 28
4.1.1 Limited Support for VFB communication features and BSW modules 28
4.1.2 Limited Support for RTE Events in Application Software Clusters . . 28
4.2 Applicability to car domains . . . 29

5 Dependencies to other modules 30
5.1 Binary Manifest . . . 30
5.1.1 Header File structure . . . 30
5.2 Cross Cluster Communication . . . 30
5.3 Proxy Modules . . . 30

6 Requirements Tracing 31

7 Functional specification 40
7.1 Binary Manifest . . . 40
7.1.1 Overview . . . 40
7.1.2 Logical structure of a Binary Manifest . . . 41
7.1.3 Mapping between Logical structure and Configuration structure . . . 43
7.1.4 Implementation structure of a Binary Manifest . . . 45
7.1.4.1 Implementation Overview . . . 48
7.1.4.2 Multiple Notifier Sets Introduction . . . 49
7.1.4.3 Binary Manifest Header . . . 50
7.1.4.4 Interface Descriptor Table . . . 53
7.1.4.5 Offered Interface Table . . . 58
7.1.4.6 Subscribed Interface Table . . . 58
7.1.4.7 Administrative Data . . . 59
7.1.4.8 Memory Mapping . . . 61
7.1.5 Connecting Software Clusters . . . 64
7.1.5.1 Overview . . . 64
7.1.5.2 Connecting Resources . . . 65
7.1.5.3 Handling of the `Substitution Software Cluster` . . . 65
7.1.5.4 Multiple Notifier Sets . . . 66
7.1.5.5 Unconnected Resources . . . 68
7.1.5.6 Disabling of the On-board Software Cluster Connection . . . 68
7.1.5.7 Errors during software cluster connection . . . 69

7.1.6 Software Cluster Binary Manifest Descriptor . . . 71
7.1.7 Error Classification . . . 71
7.1.7.1 Development Errors . . . 71
7.1.7.2 Runtime Errors . . . 71
7.1.7.3 Production Errors . . . 72
7.2 Software Cluster Base Configuration Check . . . 72
7.3 Cross Cluster Communication . . . 73
7.3.1 Overview . . . 73
7.3.2 General Requirements . . . 74
7.3.2.1 Cross Software Cluster Communication Plug-In . . . 74
7.3.2.2 Cross Cluster Communication and Binary Manifest . . . 76
7.3.2.3 Cross Cluster Communication Base Socket . . . 89
7.3.2.4 Cross Cluster Communication and McSupport Data . . . 90
7.3.3 Sender Receiver Communication . . . 92
7.3.3.1 Restrictions on VFB communication features . . . 92
7.3.3.2 Data Concistency . . . 93
7.3.3.3 Transmission . . . 93
7.3.3.4 Reception . . . 96
7.3.4 NvData communication . . . 100
7.3.4.1 Restrictions on VFB communication features . . . 100
7.3.5 Client Server Communication . . . 101
7.3.5.1 General . . . 101
7.3.5.2 Timeout . . . 101
7.3.5.3 Scheduling of main functions . . . 103
7.3.5.4 Buffering . . . 104
7.3.5.5 Response to Request Mapping . . . 106
7.3.5.6 Client Side . . . 107
7.3.5.7 Server Side . . . 109
7.3.6 Modes Communication . . . 110
7.3.6.1 General principles . . . 110
7.3.6.2 Software Cluster providing a mode . . . 111
7.3.6.3 Host Software Cluster requiring a mode . . . 116
7.3.6.4 Application Software Cluster requiring a mode . . . 117
7.3.6.5 Initialization . . . 119
7.3.7 Trigger Communication . . . 119
7.3.7.1 General principles . . . 119
7.3.7.2 Software Cluster providing a trigger . . . 120
7.3.7.3 Host Software Cluster requiring a trigger . . . 124
7.3.7.4 Application Software Cluster requiring a trigger . . . 124
7.3.8 Parameter Communication . . . 127
7.3.9 Error Classification . . . 128
7.3.9.1 Development Errors . . . 128
7.3.9.2 Runtime Errors . . . 128

7.3.9.3 Production Errors . . . 128
7.3.10 Initialization of Xcc . . . 128
7.4 Proxy Modules . . . 129
7.4.1 Overview . . . 129
7.4.2 Abstract Proxy Module Pattern . . . 131
7.4.2.1 General Proxy functionality . . . 131
7.4.2.2 Partitions . . . 132
7.4.2.3 Unconnected Service Resources . . . 134
7.4.2.4 Ecu Configuration Principles . . . 135
7.4.2.5 Proxy Modules and Binary Manifest . . . 136
7.4.2.6 Proxy Modules and Base Sockets . . . 136
7.4.3 Specific Proxy Module Requirements . . . 137
7.4.3.1 OS Proxy . . . 137
7.4.3.2 NvM Proxy . . . 153
7.4.3.3 Com Proxy . . . 169
7.4.3.4 Dcm Proxy . . . 204
7.4.3.5 Dem Proxy . . . 230
7.4.3.6 FiM Proxy . . . 241
7.4.4 Error Classification . . . 245
7.4.4.1 Development Errors . . . 245
7.4.4.2 Runtime Errors . . . 247
7.4.4.3 Production Errors . . . 247
7.5 Standardized Service Resources . . . 247
7.5.1 Software Cluster Base Configuration Check . . . 248
7.5.2 Cross Cluster Communication . . . 249
7.5.3 OS Proxy . . . 249
7.5.4 NvM Proxy . . . 251
7.5.5 Com Proxy . . . 252
7.5.6 Dcm Proxy . . . 252
7.5.7 Dem Proxy . . . 254
7.5.8 FiM Proxy . . . 255
7.5.9 WdgM . . . 255
7.6 Common Functions . . . 256
7.6.1 Error Classification . . . 256
7.6.1.1 Development Errors . . . 256
8 API specification 257
8.1 Imported types . . . 257
8.2 Type definitions . . . 258
8.2.1 Binary Manifest . . . 258
8.2.1.1 SwCluC_BManif_SwClusterIdType . . . 258
8.2.1.2 SwCluC_BManif_MachineIdType . . . 258
8.2.1.3 SwCluC_BManif_ConCtrlType . . . 258
8.2.1.4 SwCluC_BManif_ResourcePropertiesType . . . 259

8.2.1.5 SwCluC_BManif_ResourceTypeIdType . . . 259
8.2.1.6 SwCluC_BManif_GlobalResourceIdType . . . 260
8.2.1.7 SwCluC_BManif_ResourceGuardValueType . . . 260
8.2.1.8 SwCluC_BManif_TableIndexType . . . 260
8.2.1.9 SwCluC_BManif_HandleIndexType . . . 261
8.2.1.10 SwCluC_BManif_VoidFncPtrType . . . 261
8.2.1.11 SwCluC_BManif_HandleType . . . 261
8.2.1.12 SwCluC_BManif_HeaderType . . . 262
8.2.2 Cross Cluster Communication . . . 264
8.2.3 ProxyModules . . . 264
8.3 Function definitions . . . 264
8.3.1 BinaryManifest . . . 265
8.3.1.1 API Principles . . . 265
8.3.1.2 SwCluC_BManif_GetHandle . . . 268
8.3.1.3 SwCluC_BManif_GetHandle . . . 269
8.3.1.4 SwCluC_BManif_GetConSwClusterId . . . 270
8.3.1.5 SwCluC_BManif_GetConSwClusterId . . . 271
8.3.1.6 SwCluC_BManif_GetNoOfHandleSets . . . 272
8.3.1.7 SwCluC_BManif_GetNoOfHandleSets . . . 272
8.3.1.8 SwCluC_BManif_GetValidityMarker . . . 273
8.3.1.9 SWCLUC_BMANIF_NO_OF_ENTRIES . . . 274
8.3.1.10 SWCLUC_BMANIF_NO_OF_NOTIFIER_SETS . . . 274
8.3.2 Cross Cluster Communication . . . 274
8.3.2.1 SwCluC_Xcc_Init1 . . . 275
8.3.2.2 SwCluC_Xcc_Init2 . . . 275
8.3.3 Proxy Modules . . . 276
8.3.3.1 Os Proxy . . . 276
8.3.3.2 NvM Proxy . . . 277
8.3.3.3 Com Proxy . . . 278
8.3.3.4 Dcm Proxy . . . 288
8.3.3.5 Dem Proxy . . . 288
8.3.3.6 FiM Proxy . . . 289
8.3.4 Common Functions . . . 289
8.4 Scheduled functions . . . 290
8.4.1 Binary Manifest . . . 290
8.4.2 Cross Cluster Communication Scheduled functions . . . 290
8.4.2.1 SwCluC_Xcc_MainFunction . . . 290
8.4.3 Proxy Modules Scheduled functions . . . 290
8.5 Expected interfaces . . . 291
8.5.1 Mandatory interfaces . . . 291
8.5.1.1 Binary Manifest . . . 291
8.5.1.2 Cross Cluster Communication . . . 291
8.5.1.3 Proxy Modules . . . 291

8.5.2 Optional interfaces . . . 291
8.5.2.1 Binary Manifest . . . 291
8.5.2.2 Cross Cluster Communication . . . 291
8.5.2.3 Proxy Modules . . . 291
8.5.3 Configurable interfaces . . . 293
8.5.3.1 Binary Manifest . . . 293
8.5.3.2 Cross Cluster Communication . . . 293
8.5.3.3 Proxy Modules . . . 294
8.5.4 Callout interfaces . . . 294
8.6 Service Interfaces . . . 294
8.6.1 Binary Manifest . . . 294
8.6.2 Cross Cluster Communication . . . 294
8.6.3 Proxy Modules . . . 294
9 Sequence diagrams 295
10 Configuration specification 296
10.1 How to read this chapter . . . 296
10.2 Containers and configuration parameters . . . 296
10.2.1 Module Configuration . . . 297
10.2.2 General configuration parameters . . . 298
10.2.3 Software Cluster Definition . . . 300
10.2.4 Software Cluster Base Configuration Check . . . 304
10.2.5 Binary Manifest . . . 307
10.2.5.1 Provide Resource Entry Group . . . 309
10.2.5.2 Provide Resource Entry . . . 311
10.2.5.3 Require Resource Entry Group . . . 314
10.2.5.4 Require Resource Entry . . . 315
10.2.5.5 Resource Type . . . 318
10.2.6 Cross Cluster Communication . . . 322
10.2.6.1 Cross Cluster Communication Base Socket . . . 324
10.2.6.2 Cross Cluster Communication Rte Plug-in Props . . . 326
10.2.6.3 Cross Cluster Communication Main Function . . . 328
10.2.7 Proxy Modules . . . 330
10.2.7.1 NvM Proxy . . . 334
10.2.7.2 Os Proxy . . . 342
10.2.7.3 Com Proxy . . . 365
10.2.7.4 Dcm Proxy . . . 381
10.2.7.5 Dem Proxy . . . 392
10.2.7.6 FiM Proxy . . . 398
10.3 Published Information . . . 404
A Not applicable requirements 405
B Referenced Meta Classes 407

C Referenced ECUC Configuration Parameters 492
C.1 EcuC . . . 492
C.1.1 EcucPartition . . . 492
C.1.2 Pdu . . . 493
C.1.3 EcucPduDedicatedPartition . . . 497
C.2 RTE . . . 498
C.2.1 RteEventToTaskMapping . . . 498
C.2.2 RteInitializationRunnableBatch . . . 504
C.3 Os . . . 504
C.3.1 OsAlarm . . . 504
C.3.2 OsApplication . . . 505
C.3.3 OsCounter . . . 509
C.3.4 OsEvent . . . 511
C.3.5 OsResource . . . 512
C.3.6 OsScheduleTable . . . 513
C.3.7 OsScheduleTableExpiryPoint . . . 515
C.3.8 OsSpinlock . . . 516
C.3.9 OsTask . . . 517
C.4 NvM . . . 520
C.4.1 NvMBlockDescriptor . . . 520
C.4.2 NvMInitBlockCallback . . . 534
C.4.3 NvMSingleBlockCallback . . . 535
C.4.4 NvMCommon . . . 536
C.5 Dcm . . . 543
C.5.1 DcmDspDid . . . 543
C.5.2 DcmDspDidInfo . . . 546
C.5.3 DcmDspData . . . 547
C.5.4 DcmDspDataInfo . . . 555
C.5.5 DcmDspPidService01 . . . 556
C.5.6 DcmDspDidRead . . . 559
C.5.7 DcmDspDidWrite . . . 560
C.5.8 DcmDspDidControl . . . 562
C.5.9 DcmDspRoutine . . . 565
C.6 Dem . . . 568
C.6.1 DemEventParameter . . . 568
C.6.2 DemDataElementClass . . . 575
C.7 FiM . . . 576
C.7.1 FiMFID . . . 576
C.8 Com . . . 577
C.8.1 ComMainFunctionTx . . . 577
C.8.2 ComMainFunctionRx . . . 579
C.8.3 ComGroupSignal . . . 580
C.8.4 ComIPdu . . . 585

C.8.5 ComSignal . . . 590
C.8.6 ComUserModule . . . 594
C.8.7 ComUserUriDefSet . . . 594
C.8.8 ComSignalGroup . . . 595
C.9 WdgM . . . 596
C.9.1 WdgMTransitionProxy . . . 596
C.9.2 WdgMBaseSocket . . . 597
D Referenced C-API 599
D.1 RTE . . . 599
D.1.1 RTE Lifecycle API Reference . . . 599
D.1.1.1 Rte_Init . . . 599
D.1.1.2 Rte_Start . . . 599
D.1.2 RTE RIPS API Reference . . . 600
D.1.2.1 Rte_Rips_DataIsUpdated . . . 600
D.1.2.2 Rte_Rips_DRead . . . 600
D.1.2.3 Rte_Rips_DataIsUpdated_EventActivation . . . 600
D.1.2.4 Rte_Rips_Feedback . . . 601
D.1.2.5 Rte_Rips_Invoke . . . 602
D.1.2.6 Rte_Rips_InvocationHandler . . . 602
D.1.2.7 Rte_Rips_Prm . . . 603
D.1.2.8 Rte_Rips_Read . . . 603
D.1.2.9 Rte_Rips_ReturnResult . . . 604
D.1.2.10 Rte_Rips_Start . . . 604
D.1.2.11 Rte_Rips_Stop . . . 605
D.1.2.12 Rte_Rips_SchM_Deinit . . . 605
D.1.2.13 Rte_Rips_SchM_Init . . . 606
D.1.2.14 Rte_Rips_SwitchNotificationStatusType . . . 606
D.1.2.15 Rte_Rips_Switch . . . 607
D.1.2.16 Rte_Rips_DequeueModeSwitch . . . 607
D.1.2.17 Rte_Rips_Trigger . . . 608
D.1.2.18 Rte_Rips_Write . . . 608
D.1.3 RTE API Reference . . . 608
D.2 OS . . . 609
D.3 NvM . . . 610
D.3.1 NvM_CancelJobs . . . 610
D.3.2 NvM_EraseNvBlock . . . 610
D.3.3 NvM_GetDataIndex . . . 611
D.3.4 NvM_GetErrorStatus . . . 611
D.3.5 NvM_InvalidateNvBlock . . . 612
D.3.6 NvM_ReadBlock . . . 612
D.3.7 NvM_ReadPRAMBlock . . . 613
D.3.8 NvM_RestoreBlockDefaults . . . 613
D.3.9 NvM_RestorePRAMBlockDefaults . . . 614

D.3.10 NvM_SetBlockProtection . . . 614
D.3.11 NvM_SetDataIndex . . . 615
D.3.12 NvM_SetRamBlockStatus . . . 615
D.3.13 NvM_WriteBlock . . . 616
D.3.14 NvM_WritePRAMBlock . . . 616
D.3.15 NvM_SingleBlockCallbackFunction . . . 617
D.3.16 NvM_InitBlockCallbackFunction . . . 617
D.3.17 NvM_ReadRamBlockFromNvm . . . 618
D.3.18 NvM_WriteRamBlockToNvm . . . 618
D.3.19 NvM_ReadAll . . . 619
D.3.20 NvM_WriteAll . . . 619
D.4 Com . . . 620
D.4.1 APIs to handle ComSignals / ComSignalGroups . . . 620
D.4.1.1 Com_SendSignal . . . 620
D.4.1.2 Com_SendDynSignal . . . 620
D.4.1.3 Com_SendSignalGroup . . . 621
D.4.1.4 Com_SendSignalGroupArray . . . 621
D.4.1.5 Com_SendSignalWithMetaData . . . 622
D.4.1.6 Com_SendDynSignalWithMetaData . . . 623
D.4.1.7 Com_SendSignalGroupWithMetaData . . . 623
D.4.1.8 Com_SendSignalGroupArrayWithMetaData . . . 624
D.4.1.9 Com_ReceiveSignal . . . 625
D.4.1.10 Com_ReceiveDynSignal . . . 625
D.4.1.11 Com_ReceiveSignalGroup . . . 626
D.4.1.12 Com_ReceiveSignalGroupArray . . . 626
D.4.1.13 Com_ReceiveSignalWithMetaData . . . 627
D.4.1.14 Com_ReceiveDynSignalWithMetaData . . . 628
D.4.1.15 Com_ReceiveSignalGroupWithMetaData . . . 628
D.4.1.16 Com_ReceiveSignalGroupArrayWithMetaData . . . 629
D.4.1.17 Com_InvalidateSignal . . . 629
D.4.1.18 Com_InvalidateSignalGroup . . . 630
D.4.1.19 ComUser_CbkRxAck . . . 630
D.4.1.20 ComUser_CbkInv . . . 631
D.4.1.21 ComUser_CbkRxTOut . . . 631
D.4.1.22 ComUser_CbkTxAck . . . 632
D.4.1.23 ComUser_CbkTxErr . . . 632
D.4.1.24 ComUser_CbkTxTOut . . . 633
D.4.2 APIs to handle ComIPdus . . . 634
D.4.2.1 Com_Transmit . . . 634
D.4.2.2 ComUser_ComCbkTriggerTransmit . . . 634
D.4.2.3 ComUser_ComCbkRxIndication . . . 635
D.4.2.4 ComUser_ComCbkTxConfirmation . . . 635
D.4.2.5 ComUser_ComCbkCopyRxData . . . 636

D.4.2.6 ComUser_ComCbkStartOfReception . . . 636
D.4.2.7 ComUser_ComCbkTpRxIndication . . . 637
D.4.2.8 ComUser_ComCbkTpTxConfirmation . . . 638
D.4.2.9 ComUser_ComCbkCopyTxData . . . 638
D.5 Dcm . . . 639
D.5.1 DataServices . . . 639
D.5.1.1 ReadData . . . 639
D.5.1.2 WriteData . . . 641
D.5.1.3 ConditionCheckRead . . . 642
D.5.1.4 ReadDataLength . . . 642
D.5.1.5 GetScalingInformation . . . 643
D.5.1.6 ReturnControlToECU . . . 645
D.5.1.7 ResetToDefault . . . 645
D.5.1.8 FreezeCurrentState . . . 647
D.5.1.9 ShortTermAdjustment . . . 648
D.5.2 RoutineServices . . . 650
D.5.2.1 Xxx_Start Operation . . . 650
D.5.2.2 Xxx_StartConfirmation Operation . . . 651
D.5.2.3 Xxx_Stop Operation . . . 651
D.5.2.4 Xxx_StopConfirmation Operation . . . 652
D.5.2.5 Xxx_RequestResults Operation . . . 652
D.5.2.6 Xxx_RequestResultsConfirmation Operation . . . 653
D.5.3 ServiceRequestNotification . . . 654
D.5.3.1 Indication . . . 654
D.5.3.2 Confirmation . . . 655
D.6 Dem . . . 655
D.6.1 Dem_ClearPrestoredFreezeFrame . . . 655
D.6.2 Dem_PrestoreFreezeFrame . . . 656
D.6.3 Dem_ResetEventDebounceStatus . . . 656
D.6.4 Dem_ResetEventStatus . . . 657
D.6.5 Dem_SetEventDisabled . . . 657
D.6.6 Dem_SetEventStatus . . . 658
D.6.7 Dem_GetDTCOfEvent . . . 659
D.6.8 Dem_GetDebouncingOfEvent . . . 659
D.6.9 Dem_GetEventAvailable . . . 660
D.6.10 Dem_GetEventExtendedDataRecordEx . . . 660
D.6.11 Dem_GetEventFreezeFrameDataEx . . . 661
D.6.12 Dem_GetEventUdsStatus . . . 662
D.6.13 Dem_GetFaultDetectionCounter . . . 662
D.6.14 Dem_GetMonitorStatus . . . 663
D.6.15 Dem_SetEventStatusWithMonitorData . . . 663
D.6.16 Dem_SetWIRStatus . . . 664
D.6.17 <Module>_DemGetFaultDetectionCounter<ForEvent> . . . 664

D.6.18 <Module>_DemInitMonitorFor<EventName> . . . 665
D.6.19 <Module>_DemTriggerOnMonitorStatus . . . 665
D.6.20 <Module>_DemClearEventAllowed<ForCondition> . . . 666
D.6.21 <Module>_DemTriggerOnEventData . . . 666
D.6.22 <Module>_DemTriggerOnEventUdsStatus . . . 667
D.7 FiM . . . 667
D.7.1 FiM_GetFunctionPermission . . . 667
D.7.2 FiM_SetFunctionAvailable . . . 668
E Referenced Service Interfaces 669
E.1 Os . . . 669
E.2 NvM . . . 669
E.2.1 NvM_BlockIdType . . . 669
E.2.2 NvM_BlockRequestType . . . 669
E.2.3 NvM_InitBlockRequestType . . . 670
E.2.4 NvM_RequestResultType . . . 670
E.2.5 NvMService . . . 671
E.2.6 NvMAdmin . . . 676
E.2.7 NvMNotifyJobFinished . . . 676
E.2.8 NvMNotifyInitBlock . . . 677
E.2.9 NvMMirror . . . 677
E.2.10 PS_{Block} . . . 678
E.2.11 PAdmin_{Block} . . . 679
E.2.12 PNJF_{Block} . . . 679
E.2.13 PNIB_{Block} . . . 680
E.2.14 PM_{Block} . . . 680
E.3 Dcm . . . 681
E.3.1 Implementation Data Types . . . 681
E.3.1.1 Dcm_OpStatusType . . . 681
E.3.1.2 Dcm_ConfirmationStatusType . . . 681
E.3.1.3 Dcm_NegativeResponseCodeType . . . 683
E.3.1.4 Dcm_DataElementType_{Data}Type . . . 687
E.3.1.5 Dcm_DataElementType_{Data}ArrayType . . . 687
E.3.1.6 Dcm_DataElementType_{Data}PrimitiveType . . . 689
E.3.1.7 Dcm_DataArrayTypeUint8_{Data}Type . . . 690
E.3.1.8 {DID}_Struct_DataType . . . 690
E.3.1.9 Dcm_RangeArray_{Range}Type . . . 691
E.3.1.10 Dcm_ScalingInfoArray_{Data}Type . . . 691
E.3.1.11 Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType . . . 692
E.3.1.12 Dcm_RequestDataIn_{Routine}_{Signal}PrimitiveType . . . 693
E.3.1.13 Dcm_RequestDataOut_{Routine}_{Signal}Type . . . 694
E.3.1.14 Dcm_RequestDataIn_{Routine}_{Signal}Type . . . 694
E.3.1.15 Dcm_RequestDataOut_{Routine}_{Signal}ArrayType . . . 695
E.3.1.16 Dcm_RequestDataIn_{Routine}_{Signal}ArrayType . . . 696

E.3.1.17 Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type . . 697
E.3.1.18 Dcm_RequestFlexibleInArrayData_{Routine}_{Signal}Type . . . 697
E.3.1.19 Dcm_StartDataIn_{Routine}_{Signal}PrimitivType 698
E.3.1.20 Dcm_StartDataIn_{Routine}_{Signal}Type 699
E.3.1.21 Dcm_StartDataIn_{Routine}_{Signal}ArrayType 699
E.3.1.22 Dcm_StartDataOut_{Routine}_{Signal}PrimitivType 700
E.3.1.23 Dcm_StartDataOut_{Routine}_{Signal}Type 701
E.3.1.24 Dcm_StartDataOut_{Routine}_{Signal}ArrayType 702
E.3.1.25 Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type 703
E.3.1.26 Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type 703
E.3.1.27 Dcm_StopDataIn_{Routine}_{Signal}PrimitivType 703
E.3.1.28 Dcm_StopDataIn_{Routine}_{Signal}Type 704
E.3.1.29 Dcm_StopDataIn_{Routine}_{Signal}ArrayType 705
E.3.1.30 Dcm_StopDataOut_{Routine}_{Signal}PrimitivType 706
E.3.1.31 Dcm_StopDataOut_{Routine}_{Signal}Type 707
E.3.1.32 Dcm_StopDataOut_{Routine}_{Signal}ArrayType 708
E.3.1.33 Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type 709
E.3.1.34 Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type 709
E.3.1.35 Dcm_KeyArray_{SecurityLevel}Type 709
E.3.1.36 Dcm_SeedArray_{SecurityLevel}Type 710
E.3.1.37 Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type . . 710
E.3.1.38 Dcm_RequestDataArrayType 711
E.3.1.39 Dcm_ControlMask_{DID}Type 711
E.3.1.40 Dcm_InputOutputControlParameterType 712
E.3.1.41 Dcm_IOOperationRequest_{DID}Type 712
E.3.1.42 Dcm_IOOperationResponseType 713
E.3.1.43 Dcm_ResponseDataArrayType 713
E.3.1.44 Dcm_ControlMask_{Data}ArrayType 714
E.3.1.45 Dcm_Cemr_{DID}Type 714
E.3.2 Sender-Receiver-Interfaces 714
E.3.2.1 DataServices_{DID} 715
E.3.2.2 DataServices_{Data} 715
E.3.2.3 IOControlRequest_{DID} 716
E.3.2.4 IOControlResponse_{DID} 716
E.3.3 NvDataInterface 717
E.3.3.1 DataServices_{DID} 717
E.3.4 Client-Server-Interfaces 717
E.3.4.1 DataServices_{Data} 717
E.3.4.2 RoutineServices_{RoutineName} 718
E.3.4.3 ServiceRequestNotification 732
E.3.5 Ports 734
E.3.5.1 DataServices_{DID} RPortProtoype 734
E.3.5.2 DataServices_{DID} PPortProtoype 735

E.3.5.3 DataServices_{DID} PRPortPrototype . . . 735
E.3.5.4 DataServices_{Data} RPortPrototype . . . 735
E.3.5.5 DataServices_{Data} RPortPrototype . . . 736
E.3.5.6 DataServices_{Data} PPortPrototype . . . 736
E.3.5.7 DataServices_{Data} PRPortPrototype . . . 736
E.3.5.8 IOControlRequest_{DID} . . . 737
E.3.5.9 IOControlResponse_{DID} . . . 737
E.3.5.10 ServiceRequestManufacturerNotification_{Name} . . . 737
E.3.5.11 ServiceRequestSupplierNotification_{Name} . . . 738
E.3.5.12 RoutineServices_{RoutineName} . . . 738
E.4 Dem . . . 738
E.4.1 Implementation Data Types . . . 738
E.4.1.1 Dem_DebouncingStateType . . . 738
E.4.1.2 Dem_DTCOriginType . . . 739
E.4.1.3 Dem_EventIdType . . . 739
E.4.1.4 Dem_EventStatusType . . . 740
E.4.1.5 Dem_DebounceResetStatusType . . . 740
E.4.1.6 Dem_DTCFormatType . . . 741
E.4.1.7 Dem_InitMonitorReasonType . . . 741
E.4.1.8 Dem_MaxDataValueType . . . 742
E.4.1.9 Dem_MonitorDataType . . . 742
E.4.1.10 Dem_MonitorStatusType . . . 742
E.4.1.11 Dem_UdsStatusByteType . . . 743
E.4.2 Client-Server-Interfaces . . . 743
E.4.2.1 CallbackClearEventAllowed . . . 743
E.4.2.2 CallbackEventDataChanged . . . 744
E.4.2.3 CallbackEventUdsStatusChanged . . . 744
E.4.2.4 CallbackGetFaultDetectCounter . . . 745
E.4.2.5 CallbackInitMonitorForEvent . . . 746
E.4.2.6 CallbackMonitorStatusChange . . . 746
E.4.2.7 DiagnosticInfo . . . 746
E.4.2.8 DiagnosticMonitor . . . 750
E.4.2.9 DiagnosticMonitor_MonitorData . . . 751
E.4.2.10 EventStatus . . . 752
E.4.3 Ports . . . 753
E.4.3.1 Event . . . 753
E.4.3.2 DiagnosticMonitor_MonitorData . . . 753
E.4.3.3 EventStatus . . . 754
E.4.3.4 EventInfo . . . 754
E.4.3.5 CBClrEvt . . . 754
E.4.3.6 CBDataEvt . . . 755
E.4.3.7 CBFaultDetectCtr . . . 755
E.4.3.8 CBInitEvt . . . 755

E.4.3.9 CBEventUdsStatusChanged . . . 756
E.4.3.10 CBMonitorStatusChanged . . . 756
E.5 FiM . . . 756
E.5.1 Implementation Data Types . . . 756
E.5.1.1 FiM_FunctionIdType . . . 756
E.5.2 Client-Server-Interfaces . . . 757
E.5.2.1 FiM_FunctionInhibition . . . 757
E.5.2.2 FiM_ControlFunctionAvailable . . . 758
E.5.3 Ports . . . 758
E.5.3.1 FiM_FunctionInhibition . . . 758
E.5.3.2 FiM_ControlFunctionAvailable . . . 759
F Changes History 760
F.1 Changes in R20-11 compared to R19-11 . . . 760
F.1.1 Added Specification Items in R20-11 . . . 760
F.1.2 Changed Specification Items in R20-11 . . . 762
F.1.3 Deleted Specification Items in R20-11 . . . 762
F.2 Changes in R21-11 compared to R20-11 . . . 763
F.2.1 Added Specification Items in R21-11 . . . 763
F.2.2 Changed Specification Items in R21-11 . . . 765
F.2.3 Deleted Specification Items in R21-11 . . . 765
F.3 Changes in R22-11 compared to R21-11 . . . 766
F.3.1 Added Specification Items in R22-11 . . . 766
F.3.2 Changed Specification Items in R22-11 . . . 766
F.3.3 Deleted Specification Items in R22-11 . . . 768
F.4 Changes in R23-11 compared to R22-11 . . . 768
F.4.1 Added Specification Items in R23-11 . . . 768
F.4.2 Changed Specification Items in R23-11 . . . 768
F.4.3 Deleted Specification Items in R23-11 . . . 769
F.4.4 Added Constraints in R23-11 . . . 769
F.4.5 Changed Constraints in R23-11 . . . 769
F.4.6 Deleted Constraints in R23-11 . . . 769
F.5 Changes in R24-11 compared to R23-11 . . . 769
F.5.1 Added Specification Items in R24-11 . . . 769
F.5.2 Changed Specification Items in R24-11 . . . 769
F.5.3 Deleted Specification Items in R24-11 . . . 769
F.5.4 Added Constraints in R24-11 . . . 771
F.5.5 Changed Constraints in R24-11 . . . 772
F.5.6 Deleted Constraints in R24-11 . . . 772
F.6 Changes in R25-11 compared to R24-11 . . . 772
F.6.1 Added Specification Items in R25-11 . . . 772
F.6.2 Changed Specification Items in R25-11 . . . 772
F.6.3 Deleted Specification Items in R25-11 . . . 773
F.6.4 Added Constraints in R25-11 . . . 773

F.6.5 Changed Constraints in R25-11 . . . 774
F.6.6 Deleted Constraints in R25-11 . . . 774
F.7 Changes in R26-11 compared to R25-11 . . . 774
F.7.1 Added Specification Items in R26-11 . . . 774
F.7.2 Changed Specification Items in R26-11 . . . 774
F.7.3 Deleted Specification Items in R26-11 . . . 774
F.7.4 Added Constraints in R26-11 . . . 774
F.7.5 Changed Constraints in R26-11 . . . 774
F.7.6 Deleted Constraints in R26-11 . . . 775

Known Limitations

- Not all VFB communication features are supported
- The Proxy Modules are not specified for all BSW Services
- No run-time separation is defined

Additional limitations are described in document [1]

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module Software Cluster Connection.

2 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to Requirements on Software Cluster Connection that are not included in the AUTOSAR Glossary [2].

Abbreviation / Acronym:	Description:
SwCluC	Software Cluster Connection

Table 2.1: Acronyms and Abbreviations

Terms:	Description:
Software Cluster	A Software Cluster groups all AUTOSAR artifacts that are relevant to deploy software on a machine. The full definition is given in document [2]
Software Cluster Connection	The Software Cluster Connection is the BSW module that provides the features to • connect the Binary Objects deployed on the same machine • substitute not locally-available BSW modules in an Application Software Cluster, whose interfaces are required for the integrated SW, by so called Proxy Modules. • implement the VFB communication features between Software Clusters together with RTE by means of an RTE Implementation Plug-In
machine	A machine consists of a set of computing resources - such as microcontroller cores, memory or peripheral (e.g. communication) devices - and has the ability to execute software applications. The representation of a machine in the AUTOSAR Classic Platform could be done with an EcuInstance, but note that this semantic is currently in clarification. Further information is given in document [2].
binary-identical	Bit for Bit identical
Binary Object	A set of files, which contains the binary executable code and data. This binary executable code and data will not be modified again, before programming it on the target ECU.
Binary Manifest	The Binary Manifest is the well-defined interface of the Software Cluster's Binary Object, providing the meta information of a resources and information - so called handles - to access such a resource.
Application Software Cluster	A Software Cluster that mainly contains software components, and only selected BSW modules (e.g. a Service module, transformers, e.t.c.)
Host Software Cluster	The single Software Cluster that contains the major part of the BSW, and especially the micro controller dependent lower layer BSW Modules, e.g. OS and MCAL.
Substitution Software Cluster	The single Software Cluster that can override the provided resources of other Software Clusters for bug fixing purpose.
Resource Provider	Is a Software Cluster providing a resource for one or several other Software Cluster(s). A resource, in this context, can be anything that is required to operate the software.

Terms:	Description:
Resource Requester	Is a Software Cluster requiring a resource from another Software Cluster. A resource, in this context, can be anything that is required to operate the software.
Proxy Module	A Proxy Module substitutes a BSW module in an Application Software Cluster. A Proxy module itself is split into High Proxy Module and Low Proxy Module. The High Proxy Module provides dedicated interfaces for modules in higher layers or same layer, and the functionality to connect them via the Binary Manifest to the Low Proxy Module in the Host Software Cluster.
High Proxy Module	The High Proxy Module respectively High Proxy is the part of the Proxy Module residing in an Application Software Cluster.
Low Proxy Module	The Low Proxy Module respectively Low Proxy is the part of the Proxy Module residing in the Host Software Cluster.
Os Proxy	The Proxy Module for Os.
Os High Proxy	A type of Proxy Module implementing Os APIs in the Application Software Cluster.
Os Low Proxy	A type of Proxy Module implementing an Os abstraction in the Host Software Cluster.
NvM Proxy	The Proxy Module for NvM.
NvM High Proxy	A type of Proxy Module substituting the NVRAM Manager in the Application Software Cluster.
NvM Low Proxy	A type of Proxy Module connecting the NvM High Proxys module to the NVRAM Manager in the Host Software Cluster.
Com Proxy	The Proxy Module for Com.
Com High Proxy	A type of Proxy Module implementing default Com (Callback) APIs in the Application Software Cluster.
Com Low Proxy	A type of Proxy Module connecting Com High Proxys module to the Com in the Host Software Cluster.
Dcm Proxy	The Proxy Module for Dcm.
Dcm High Proxy	A type of Proxy Module substituting the Dcm in the Application Software Cluster.
Dcm Low Proxy	A type of Proxy Module connecting Dcm High Proxys module to the Dcm in the Host Software Cluster.
Dem Proxy	The Proxy Module for Dem.
Dem High Proxy	A type of Proxy Module substituting the Dem in the Application Software Cluster.
Dem Low Proxy	A type of Proxy Module connecting Dem High Proxys module to the Dem in the Host Software Cluster.
FiM Proxy	The Proxy Module for FiM.
FiM High Proxy	A type of Proxy Module substituting the FiM in the Application Software Cluster.
FiM Low Proxy	A type of Proxy Module connecting FiM High Proxys module to the FiM in the Host Software Cluster.
Cross Cluster Communication	The Cross Cluster Communication implements the VFB communication between Software Clusters.
Software Cluster Connector	The Software Cluster Connector reads the information of the unconnected Binary Manifests and calculates the modifiable Binary Manifest parts.
Base Socket	Some general links to connect some generic infrastructure between Low Proxy Module and High Proxy Module(e.g. the set of BSW APIs).

Terms:	Description:
Dispatcher	A Dispatcher is a Runnable in the Host Software Cluster, which provides the ability to schedule either OS task bodies or functions in Application Software Clusters.
NVRAM configuration ID	NVRAM configuration ID is a unique configuration identifier of the non-volatile memory configuration.
IOC	IOC stands for Inter OS-Application Communicator. The IOC is responsible for the communication between OS-Applications and in particular for the communication crossing core or memory protection boundaries.
Service Software Component	A Service Software Component represents an AUTOSAR Service and is described with the meta-model class ServiceSwComponentType.
Diagnostic Event	A Diagnostic Event defines the atomic unit that can be handled by the Dem module. The status of a Diagnostic Event represents the result of a monitor.
RTE Implementation Plug-In	A RTE Implementation Plug-In is a part of the overall RTE implementation, which is not provided by the RTE Generator, but from an additional source (e.g. a Plug-In Generator or a manually implemented source code).
RTE Implementation Plug-In Service	A RTE Implementation Plug-In Service is a single entry point into the RTE Implementation Plug-In implementing a low level service for the RTE. For instance access to a specific buffer.
Local Software Cluster Communication Plug-In	A Local Software Cluster Communication Plug-In is an RTE Implementation Plug-In, which handles the communication locally inside a Software Cluster. This includes the Transformer handling, if a DataMapping exist for the according Communication Graph
Cross Software Cluster Communication Plug-In	A Cross Software Cluster Communication Plug-In is an RTE Implementation Plug-In that handles the communication towards other Software Clusters. This includes the Transformer handling, if intra ECU transformation is configured.
Communication Graph	The sum of all AbstractAccessPoints to elements of PortInterfaces, instantiated in PortPrototypes which are connected to each other; or the sum of all accesses from BswModuleEntitys to interface elements in a BswModuleDescriptions connected to each other.
Data Communication Graph	The sum of all VariableAccesses to VariableDataPrototypes instantiated in PortPrototypes, which are connected to each other; or the sum of all VariableAccesses to VariableDataPrototypes in the InternalBehavior; or the sum of all BswVariableAccesses to VariableDataPrototypes in BswModuleDescriptions connected to each other.
Parameter Communication Graph	The sum of all ParameterAccesses to ParameterDataPrototypes instantiated in PortPrototypes, which are connected to each other; or the sum of all ParameterAccesses to ParameterDataPrototypes in the InternalBehavior.
Client Server Communication Graph	The sum of all ServerCallPoints to operations instantiated in PortPrototypes, which are connected to each other, including the associated server runnable.

Terms:	Description:
client	A client is defined as one ClientServerOperation in one RPortPrototype of one software component instance. For the definition of the client neither the number of ServerCallPoints nor RunnableEntity accesses to the ServerCallPoint are relevant. A software component instance can appear as several clients to the same server if it defines ServerCallPoints for several PortPrototypes of the same PortInterface's ClientServerOperation.
server	A server is defined as one RunnableEntity which is the target of an OperationInvokedEvent. Call serialization is on activation of RunnableEntity.
server runnable	A server that is triggered by an OperationInvokedEvent. It has a mixed behavior between a runnable and a function call. In certain situations, RTE can implement the client server communication as a simple function call.
ASCR runnable	A *Runnable Entity* that is triggered by at least one AsynchronousServerCallReturnsEvent.
Trigger Communication Graph	The sum of all ExternalTriggeringPoints for triggers instantiated in PortPrototypes, which are connected to each other, including the associated triggered runnable.
Mode Communication Graph	The sum of all ModeAccessPoints and ModeSwitchPoints to ModeDeclarationGroupPrototypes instantiated in PortPrototypes, which are connected to each other; or the sum of all managedModeGroups and accessedModeGroups to ModeDeclarationGroupPrototypes in BswModuleDescriptions connected to each other.
mode manager	Entering and leaving modes is initiated by a *mode manager*. A *mode manager* is either a software component that provides a p-port typed by a ModeSwitchInterface, or a BSW module that defines in its BswModuleDescription a ModeDeclarationGroupPrototype in the role providedModeGroup.
mode switch notification	The communication of a mode switch from the mode manager to the mode user, using either the ModeSwitchInterface or providedModeGroup and requiredModeGroup ModeDeclarationGroupPrototypes.
mode switch port	The port for receiving (or sending) a mode switch notification. For this purpose, a mode switch port is typed by a ModeSwitchInterface.
mode user	An *AUTOSAR SW-C* or *AUTOSAR Basic Software Module* that depends on modes, is called a mode user. The dependency can occur through a SwcModeSwitchEvent/BswModeSwitchEvent, a ModeAccessPoint for a provided/required mode switch port, or a accessedModeGroup for an providedModeGroup/requiredModeGroup ModeDeclarationGroupPrototype.

Terms:	Description:
mode machine instance	The instances of mode machines or *ModeDeclarationGroup*s are defined by the *ModeDeclarationGroupPrototype*s of the mode managers. Since a mode switch is not executed instantaneously, The RTE or *Basic Software Scheduler* has to maintain it's own states. For each mode manager's ModeDeclarationGroupPrototype, RTE or *Basic Software Scheduler* has one state machine. This state machine is called *mode machine instance*. For all mode users of the same mode manager's *ModeDeclarationGroupPrototype*, RTE and *Basic Software Scheduler* uses the same *mode machine instance*.
on-entry ExecutableEntity	A RunnableEntity that is triggered by a SwcModeSwitchEvent with *ModeActivationKind* 'entry'; or a BswSchedulableEntity that is triggered by a BswModeSwitchEvent with *ModeActivationKind* 'entry'.
on-exit ExecutableEntity	A RunnableEntity that is triggered by a SwcModeSwitchEvent with *ModeActivationKind* 'exit'; or a BswSchedulableEntity that is triggered by a BswModeSwitchEvent with *ModeActivationKind* 'exit'.
on-transition ExecutableEntity	A RunnableEntity that is triggered by a SwcModeSwitchEvent with *ModeActivationKind* 'transition'; or a BswSchedulableEntity that is triggered by a BswModeSwitchEvent with *ModeActivationKind* 'transition'.
trigger port	A PortPrototype, which is typed by an TriggerInterface
trigger sink	A *trigger sink* relies on the activation of RunnableEntity or a BswSchedulableEntity, if a particular Trigger is raised. A *trigger sink* has a dedicated require trigger port(s) and / or requiredTrigger Trigger(s) to communicate to the trigger source(s).
trigger source	A *trigger source* administrates the particular Trigger, and informs the RTE or *Basic Software Scheduler* if the Trigger is raised. A *trigger source* has dedicated provide trigger port(s) and / or releasedTrigger Trigger(s) to communicate to the trigger sink(s).
triggered ExecutableEntity	A RunnableEntity that is triggered by at least one ExternalTriggerOccurredEvent / InternalTriggerOccurredEvent; or a BswSchedulableEntity that is triggered by at least one BswExternalTriggerOccurredEvent/ BswInternalTriggerOccurredEvent. In particular cases, the Trigger Event Communication or the *Inter Runnable Triggering* is implemented by RTE or *Basic Software Scheduler* as a direct or trusted function call of the triggered ExecutableEntity, by the triggering ExecutableEntity.
triggered runnable	A RunnableEntity that is triggered at least by one ExternalTriggerOccurredEvent or InternalTriggerOccurredEvent. In particular cases, the Trigger Event Communication or the *Inter Runnable Triggering* is implemented by RTE as a direct or trusted function call of the *triggered runnable*, by the triggering runnable.
trigger notification	RunnableEntity and belonging Dispatcher informing the RTE in the Application Software Cluster about the occurrence of a trigger. This happens by calling belonging Rte_Trigger API in the Application Software Cluster's RTE.

Terms:	Description:

Table 2.2: Terms

3 Related documentation

3.1 Input documents & related standards and norms

[1] Explanation of Software Cluster Design And Integration Guideline for Classic Platform
AUTOSAR_CP_EXP_SwClusterDesignAndIntegrationGuideline

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[4] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[5] Specification of Operating System
AUTOSAR_CP_SWS_OS

[6] Specification of NVRAM Manager
AUTOSAR_CP_SWS_NVRAMManager

[7] Specification of Communication
AUTOSAR_CP_SWS_COM

[8] Specification of Diagnostic Communication Manager
AUTOSAR_CP_SWS_DiagnosticCommunicationManager

[9] Specification of Diagnostic Event Manager
AUTOSAR_CP_SWS_DiagnosticEventManager

[10] Specification of Function Inhibition Manager
AUTOSAR_CP_SWS_FunctionInhibitionManager

[11] Requirements on Software Cluster Connection module
AUTOSAR_CP_RS_SoftwareClusterConnection

[12] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[13] System Template
AUTOSAR_CP_TPS_SystemTemplate

[14] Specification of CRC Library
AUTOSAR_CP_SWS_CRCLibrary

[15] Specification of Memory Mapping
AUTOSAR_CP_SWS_MemoryMapping

[16] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[17] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[18] Guide to BSW Distribution
AUTOSAR_CP_EXP_BSWDistributionGuide

[19] Specification of Watchdog Manager
AUTOSAR_CP_SWS_WatchdogManager

[20] Specification of Platform Types for Classic Platform
AUTOSAR_CP_SWS_PlatformTypes

[21] Specification of Standard Types
AUTOSAR_CP_SWS_StandardTypes

[22] Basic Software Module Description Template
AUTOSAR_CP_TPS_BSWModuleDescriptionTemplate

[23] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate

[24] ISO 17356-3: Road vehicles – Open interface for embedded automotive applications – Part 3: OSEK/VDX Operating System (OS)

[25] ISO 17356-4: Road vehicles – Open interface for embedded automotive applications – Part 4: OSEK/VDX Communication (COM)

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for Software Cluster Connection.

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

4 Constraints and assumptions

4.1 Limitations

4.1.1 Limited Support for VFB communication features and BSW modules

The specification currently supports a limited number of BSW modules. Furthermore, for the supported BSW modules, the functionality and APIs available for Application Software Clusters is only subset of the overall functionality. In addition, the available VFB communication features are restricted.

The supported VFB features are listed in 7.3.1.

4.1.2 Limited Support for RTE Events in Application Software Clusters

Due to limitations in the specification of the Cross Software Cluster Communication Plug-Ins in Section 7.3.2.1 and in document [4] Chapter 7 *"RTE Implementation Plug-Ins Reference"*, as well as limitations in the functionality of Dispatchers in Section 7.4.3.1.5 (scheduling via periodic TimingEvent), the functionality is specified for only a subset of RTEEvents in Application Software Clusters . These are:

- TimingEvent
- InitEvent
- BackgroundEvent
- ExternalTriggerOccurredEvent
- SwcModeSwitchEvent
- DataReceivedEvent
- AsynchronousServerCallReturnsEvent
- OperationInvokedEvent

For all other kinds of RTEEvents, such as:

- DataReceiveErrorEvent
- DataSendCompletedEvent
- DataWriteCompletedEvent
- InternalTriggerOccurredEvent
- OsTaskExecutionEvent
- SwcModeManagerErrorEvent
- TransformerHardErrorEvent

the implementation is undefined by AUTOSAR.

4.2 Applicability to car domains

The specification focus on larger ECUs centralizing software functionality - so called domain or zone controllers. It assumes that the software components which are mapped to different `Software Clusters` are rather loosely coupled w.r.t. to interfaces and time domain. The software components supposed to implement mainly control loop software - usually time driven but may also react on limited numbers of sporadic events. Further information can be found in document [1].

5 Dependencies to other modules

5.1 Binary Manifest

5.1.1 Header File structure

[SWS_SwCluC_00013]

Upstream requirements: SRS_SwCluC_00006

⌈The Binary Manifest of the Software Cluster Connection shall provide the header file SwCluC_BManif.h.

⌋

[SWS_SwCluC_00014]

Upstream requirements: SRS_SwCluC_00006

⌈The header file SwCluC_BManif.h shall include all header files configured in SwCluCBManifHeaderIncludes in the containers SwCluCBManifProvideResourceEntryGroup and SwCluCBManifRequireResourceEntryGroup.⌋

5.2 Cross Cluster Communication

The Cross Cluster Communication of the Software Cluster Connection depends on the RTE [4].

5.3 Proxy Modules

The SwCluC module has following dependencies for its sub-functionalities in the Proxy Modules:

Each Low Proxy Module depends on its specific BSW module

- Os Low Proxy depends on OS [5]
- NvM Low Proxy depends on NvM [6]
- Com Low Proxy depends on Com [7]
- Dcm Low Proxy depends on Dcm [8]
- Dem Low Proxy depends on Dem [9]
- FiM Low Proxy depends on FiM [10]

6 Requirements Tracing

The following tables reference the requirements specified in [11], SRS_SoftwareClusterConnection and [12], SRS_BSWGeneral and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_SwCluC_02143] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100] [SWS_SwCluC_12200] [SWS_SwCluC_12300] [SWS_SwCluC_12400] [SWS_SwCluC_13001]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_SwCluC_02005] [SWS_SwCluC_02006] [SWS_SwCluC_02007]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_SwCluC_CONSTR_00078] [SWS_SwCluC_CONSTR_00096] [SWS_SwCluC_CONSTR_02130] [SWS_SwCluC_CONSTR_02131] [SWS_SwCluC_CONSTR_02145] [SWS_SwCluC_CONSTR_02146] [SWS_SwCluC_CONSTR_02158] [SWS_SwCluC_CONSTR_02159] [SWS_SwCluC_CONSTR_02231] [SWS_SwCluC_CONSTR_02232] [SWS_SwCluC_CONSTR_02233] [SWS_SwCluC_CONSTR_02234] [SWS_SwCluC_CONSTR_02235] [SWS_SwCluC_CONSTR_02237] [SWS_SwCluC_CONSTR_02427] [SWS_SwCluC_CONSTR_02428] [SWS_SwCluC_CONSTR_02429] [SWS_SwCluC_CONSTR_02664] [SWS_SwCluC_CONSTR_02720] [SWS_SwCluC_CONSTR_02721] [SWS_SwCluC_CONSTR_02752] [SWS_SwCluC_CONSTR_03020] [SWS_SwCluC_CONSTR_03021] [SWS_SwCluC_CONSTR_03032] [SWS_SwCluC_CONSTR_03053] [SWS_SwCluC_CONSTR_03054] [SWS_SwCluC_CONSTR_03059] [SWS_SwCluC_CONSTR_03063] [SWS_SwCluC_CONSTR_03069] [SWS_SwCluC_CONSTR_03093] [SWS_SwCluC_CONSTR_03210] [SWS_SwCluC_CONSTR_03211] [SWS_SwCluC_CONSTR_03400] [SWS_SwCluC_CONSTR_03403]
[SRS_BSW_00172]	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	[SWS_SwCluC_03170] [SWS_SwCluC_03173] [SWS_SwCluC_91002]
[SRS_BSW_00310]	API naming convention	[SWS_SwCluC_10000] [SWS_SwCluC_10001] [SWS_SwCluC_10002] [SWS_SwCluC_10003] [SWS_SwCluC_10004] [SWS_SwCluC_10005] [SWS_SwCluC_10006] [SWS_SwCluC_10007] [SWS_SwCluC_10008] [SWS_SwCluC_10011] [SWS_SwCluC_10020] [SWS_SwCluC_10021] [SWS_SwCluC_10022] [SWS_SwCluC_10023] [SWS_SwCluC_10032] [SWS_SwCluC_10033] [SWS_SwCluC_10034] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100] [SWS_SwCluC_12200] [SWS_SwCluC_12300] [SWS_SwCluC_12400] [SWS_SwCluC_13001]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02636] [SWS_SwCluC_02637] [SWS_SwCluC_02638] [SWS_SwCluC_02751] [SWS_SwCluC_03217] [SWS_SwCluC_03218] [SWS_SwCluC_03533] [SWS_SwCluC_03534]
[SRS_BSW_00327]	Error values naming convention	[SWS_SwCluC_02140] [SWS_SwCluC_02426] [SWS_SwCluC_02650] [SWS_SwCluC_02750] [SWS_SwCluC_03216] [SWS_SwCluC_09001]
[SRS_BSW_00337]	Classification of development errors	[SWS_SwCluC_02140] [SWS_SwCluC_02426] [SWS_SwCluC_02650] [SWS_SwCluC_02750] [SWS_SwCluC_03216] [SWS_SwCluC_09001]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02636] [SWS_SwCluC_02637] [SWS_SwCluC_02638] [SWS_SwCluC_02751] [SWS_SwCluC_03217] [SWS_SwCluC_03218] [SWS_SwCluC_03533] [SWS_SwCluC_03534]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100] [SWS_SwCluC_12200] [SWS_SwCluC_12300] [SWS_SwCluC_12400] [SWS_SwCluC_13001]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02636] [SWS_SwCluC_02637] [SWS_SwCluC_02638] [SWS_SwCluC_02751] [SWS_SwCluC_03217] [SWS_SwCluC_03218] [SWS_SwCluC_03533] [SWS_SwCluC_03534]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_SwCluC_91002]
[SRS_BSW_00385]	List possible error notifications	[SWS_SwCluC_02140] [SWS_SwCluC_02426] [SWS_SwCluC_02650] [SWS_SwCluC_02750] [SWS_SwCluC_03216] [SWS_SwCluC_09001]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_SwCluC_01000] [SWS_SwCluC_01002]
[SRS_BSW_00478]	Timing limits of main functions	[SWS_SwCluC_02275] [SWS_SwCluC_03096] [SWS_SwCluC_03098]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_SwCluC_02140] [SWS_SwCluC_02426] [SWS_SwCluC_02650] [SWS_SwCluC_02750] [SWS_SwCluC_03216] [SWS_SwCluC_09001]
[SRS_BSW_00487]	Errors for module initialization shall follow a naming rule	[SWS_SwCluC_02140] [SWS_SwCluC_02426] [SWS_SwCluC_02650] [SWS_SwCluC_02750] [SWS_SwCluC_03216] [SWS_SwCluC_09001]
[SRS_SwCluC_00001]	Easy target machine interpretation	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00015] [SWS_SwCluC_00016] [SWS_SwCluC_00017] [SWS_SwCluC_00018] [SWS_SwCluC_00019] [SWS_SwCluC_00020] [SWS_SwCluC_00021] [SWS_SwCluC_00022] [SWS_SwCluC_00023] [SWS_SwCluC_00024] [SWS_SwCluC_00025] [SWS_SwCluC_00026] [SWS_SwCluC_00027] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00036] [SWS_SwCluC_00037] [SWS_SwCluC_00040] [SWS_SwCluC_00041] [SWS_SwCluC_00042] [SWS_SwCluC_00056] [SWS_SwCluC_00070] [SWS_SwCluC_00072] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00085] [SWS_SwCluC_00086] [SWS_SwCluC_00088] [SWS_SwCluC_00090] [SWS_SwCluC_00095] [SWS_SwCluC_CONSTR_00073]
[SRS_SwCluC_00002]	Identification of intended connections by unique IDs	[SWS_SwCluC_00015] [SWS_SwCluC_00021] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00043]
[SRS_SwCluC_00003]	Bidirectional connections	[SWS_SwCluC_00024] [SWS_SwCluC_00026]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00004]	Connection multiplicity	[SWS_SwCluC_00027] [SWS_SwCluC_00045] [SWS_SwCluC_00046] [SWS_SwCluC_00047] [SWS_SwCluC_00088] [SWS_SwCluC_00089]
[SRS_SwCluC_00005]	Substitute Resource Providers	[SWS_SwCluC_00008] [SWS_SwCluC_00054] [SWS_SwCluC_00055] [SWS_SwCluC_CONSTR_00087]
[SRS_SwCluC_00006]	C API	[SWS_SwCluC_00002] [SWS_SwCluC_00013] [SWS_SwCluC_00014] [SWS_SwCluC_00062] [SWS_SwCluC_00063] [SWS_SwCluC_00064] [SWS_SwCluC_00065] [SWS_SwCluC_00066] [SWS_SwCluC_00067] [SWS_SwCluC_00068] [SWS_SwCluC_00069] [SWS_SwCluC_01000] [SWS_SwCluC_01002] [SWS_SwCluC_10000] [SWS_SwCluC_10001] [SWS_SwCluC_10002] [SWS_SwCluC_10003] [SWS_SwCluC_10004] [SWS_SwCluC_10005] [SWS_SwCluC_10006] [SWS_SwCluC_10007] [SWS_SwCluC_10008] [SWS_SwCluC_10009] [SWS_SwCluC_10010] [SWS_SwCluC_10011] [SWS_SwCluC_10020] [SWS_SwCluC_10021] [SWS_SwCluC_10022] [SWS_SwCluC_10023] [SWS_SwCluC_10032] [SWS_SwCluC_10033] [SWS_SwCluC_10034]
[SRS_SwCluC_00008]	Retrieve connection status and connected Software Cluster	[SWS_SwCluC_10021] [SWS_SwCluC_10023]
[SRS_SwCluC_00009]	Support missing interface partners	[SWS_SwCluC_00015] [SWS_SwCluC_00019] [SWS_SwCluC_00030] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00044]
[SRS_SwCluC_00010]	Static safeguard of Software Cluster connections	[SWS_SwCluC_00015] [SWS_SwCluC_00022] [SWS_SwCluC_03005] [SWS_SwCluC_03033] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03045] [SWS_SwCluC_03046] [SWS_SwCluC_03047]
[SRS_SwCluC_00011]	Separation of immutable memory and memory modifiable at the connection phase	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00015] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00042] [SWS_SwCluC_00070] [SWS_SwCluC_00072] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00085]
[SRS_SwCluC_00012]	direct linkage	[SWS_SwCluC_00057] [SWS_SwCluC_00058] [SWS_SwCluC_00059] [SWS_SwCluC_00060] [SWS_SwCluC_00061]
[SRS_SwCluC_00013]	Initialization with C-compiler and linker means	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00042] [SWS_SwCluC_00070] [SWS_SwCluC_00072]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00014]	Standardized persistence in memory	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00015] [SWS_SwCluC_00016] [SWS_SwCluC_00017] [SWS_SwCluC_00018] [SWS_SwCluC_00019] [SWS_SwCluC_00020] [SWS_SwCluC_00021] [SWS_SwCluC_00022] [SWS_SwCluC_00023] [SWS_SwCluC_00024] [SWS_SwCluC_00025] [SWS_SwCluC_00026] [SWS_SwCluC_00027] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00036] [SWS_SwCluC_00037] [SWS_SwCluC_00040] [SWS_SwCluC_00041] [SWS_SwCluC_00042] [SWS_SwCluC_00056] [SWS_SwCluC_00070] [SWS_SwCluC_00072] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00085] [SWS_SwCluC_00086] [SWS_SwCluC_00088] [SWS_SwCluC_00090] [SWS_SwCluC_00095] [SWS_SwCluC_00097] [SWS_SwCluC_CONSTR_00073] [SWS_SwCluC_CONSTR_00091] [SWS_SwCluC_CONSTR_00092] [SWS_SwCluC_CONSTR_00093] [SWS_SwCluC_CONSTR_00094]
[SRS_SwCluC_00100]	Cross Software Cluster Communication Plug-Ins	[SWS_SwCluC_03000] [SWS_SwCluC_03001] [SWS_SwCluC_03002] [SWS_SwCluC_03003] [SWS_SwCluC_03004] [SWS_SwCluC_03006] [SWS_SwCluC_03007] [SWS_SwCluC_03008] [SWS_SwCluC_03009] [SWS_SwCluC_03010] [SWS_SwCluC_03011] [SWS_SwCluC_03064] [SWS_SwCluC_03065] [SWS_SwCluC_03068] [SWS_SwCluC_03080] [SWS_SwCluC_03081] [SWS_SwCluC_03082] [SWS_SwCluC_03083] [SWS_SwCluC_03084] [SWS_SwCluC_03085] [SWS_SwCluC_03086] [SWS_SwCluC_03087] [SWS_SwCluC_03088] [SWS_SwCluC_03089] [SWS_SwCluC_03090] [SWS_SwCluC_03091] [SWS_SwCluC_03092] [SWS_SwCluC_03094] [SWS_SwCluC_03095] [SWS_SwCluC_03096] [SWS_SwCluC_03097] [SWS_SwCluC_03098] [SWS_SwCluC_03099] [SWS_SwCluC_03101] [SWS_SwCluC_03102] [SWS_SwCluC_03103] [SWS_SwCluC_03106] [SWS_SwCluC_03107] [SWS_SwCluC_03108] [SWS_SwCluC_03109] [SWS_SwCluC_03110] [SWS_SwCluC_03111] [SWS_SwCluC_03112] [SWS_SwCluC_03113] [SWS_SwCluC_03120] [SWS_SwCluC_03121] [SWS_SwCluC_03122] [SWS_SwCluC_03123] [SWS_SwCluC_03124] [SWS_SwCluC_03126] [SWS_SwCluC_03130] [SWS_SwCluC_03131] [SWS_SwCluC_03132] [SWS_SwCluC_03133] [SWS_SwCluC_03134] [SWS_SwCluC_03135] [SWS_SwCluC_03136] [SWS_SwCluC_03137] [SWS_SwCluC_03138] [SWS_SwCluC_03139] [SWS_SwCluC_03140] [SWS_SwCluC_03142] [SWS_SwCluC_03143] [SWS_SwCluC_03144] [SWS_SwCluC_03145] [SWS_SwCluC_03146] [SWS_SwCluC_03147] [SWS_SwCluC_03150] [SWS_SwCluC_03152] [SWS_SwCluC_03153] [SWS_SwCluC_03154] [SWS_SwCluC_03155] [SWS_SwCluC_03156] [SWS_SwCluC_03158] [SWS_SwCluC_03159] [SWS_SwCluC_03160] [SWS_SwCluC_03161] [SWS_SwCluC_03162] [SWS_SwCluC_03163] [SWS_SwCluC_03164] [SWS_SwCluC_03165] [SWS_SwCluC_03166] [SWS_SwCluC_03167] [SWS_SwCluC_03168] [SWS_SwCluC_03169] [SWS_SwCluC_03170] [SWS_SwCluC_03171] [SWS_SwCluC_03173] [SWS_SwCluC_03176] [SWS_SwCluC_03177] [SWS_SwCluC_03178] [SWS_SwCluC_03401] [SWS_SwCluC_03402] [SWS_SwCluC_03405] [SWS_SwCluC_03406] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_CONSTR_03066] [SWS_SwCluC_CONSTR_03067] [SWS_SwCluC_CONSTR_03069] [SWS_SwCluC_CONSTR_03093] [SWS_SwCluC_CONSTR_03400] [SWS_SwCluC_CONSTR_03403] [SWS_SwCluC_CONSTR_03404]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00101]	'1:n' Sender-receiver communication	[SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03064] [SWS_SwCluC_03065] [SWS_SwCluC_03068] [SWS_SwCluC_03080] [SWS_SwCluC_03081] [SWS_SwCluC_03082] [SWS_SwCluC_03084] [SWS_SwCluC_03085] [SWS_SwCluC_03101] [SWS_SwCluC_03102] [SWS_SwCluC_03103] [SWS_SwCluC_03106] [SWS_SwCluC_03107] [SWS_SwCluC_03108] [SWS_SwCluC_03109] [SWS_SwCluC_03110] [SWS_SwCluC_03111] [SWS_SwCluC_03112] [SWS_SwCluC_03113] [SWS_SwCluC_03120] [SWS_SwCluC_03121] [SWS_SwCluC_03122] [SWS_SwCluC_03123] [SWS_SwCluC_03124] [SWS_SwCluC_03126] [SWS_SwCluC_03130] [SWS_SwCluC_03131] [SWS_SwCluC_03132] [SWS_SwCluC_03133] [SWS_SwCluC_03135] [SWS_SwCluC_03136] [SWS_SwCluC_03137] [SWS_SwCluC_03138] [SWS_SwCluC_03139] [SWS_SwCluC_03140] [SWS_SwCluC_03142] [SWS_SwCluC_03143] [SWS_SwCluC_03144] [SWS_SwCluC_03156] [SWS_SwCluC_03158] [SWS_SwCluC_03159] [SWS_SwCluC_03405] [SWS_SwCluC_03406]
[SRS_SwCluC_00102]	'n:1' Sender-receiver communication	[SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044]
[SRS_SwCluC_00103]	'n:1' Client-server communication	[SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03047] [SWS_SwCluC_03083] [SWS_SwCluC_03086] [SWS_SwCluC_03087] [SWS_SwCluC_03088] [SWS_SwCluC_03089] [SWS_SwCluC_03090] [SWS_SwCluC_03091] [SWS_SwCluC_03092] [SWS_SwCluC_03094] [SWS_SwCluC_03095] [SWS_SwCluC_03096] [SWS_SwCluC_03097] [SWS_SwCluC_03098] [SWS_SwCluC_03099] [SWS_SwCluC_03134] [SWS_SwCluC_03145] [SWS_SwCluC_03146] [SWS_SwCluC_03147] [SWS_SwCluC_03150] [SWS_SwCluC_03152] [SWS_SwCluC_03153] [SWS_SwCluC_03154] [SWS_SwCluC_03155] [SWS_SwCluC_03160] [SWS_SwCluC_03161] [SWS_SwCluC_03162] [SWS_SwCluC_03163] [SWS_SwCluC_03164] [SWS_SwCluC_03165] [SWS_SwCluC_03166] [SWS_SwCluC_03167] [SWS_SwCluC_03168] [SWS_SwCluC_03169] [SWS_SwCluC_03170] [SWS_SwCluC_03171] [SWS_SwCluC_03173] [SWS_SwCluC_03176] [SWS_SwCluC_03177] [SWS_SwCluC_03178] [SWS_SwCluC_03401] [SWS_SwCluC_03402] [SWS_SwCluC_CONSTR_03093] [SWS_SwCluC_CONSTR_03403]
[SRS_SwCluC_00104]	'1:n' Mode Switch Communication	[SWS_SwCluC_03015] [SWS_SwCluC_03016] [SWS_SwCluC_03017] [SWS_SwCluC_03018] [SWS_SwCluC_03019] [SWS_SwCluC_03022] [SWS_SwCluC_03023] [SWS_SwCluC_03024] [SWS_SwCluC_03025] [SWS_SwCluC_03026] [SWS_SwCluC_03027] [SWS_SwCluC_03028] [SWS_SwCluC_03029] [SWS_SwCluC_03030] [SWS_SwCluC_03031] [SWS_SwCluC_03045] [SWS_SwCluC_03057] [SWS_SwCluC_03061] [SWS_SwCluC_03062] [SWS_SwCluC_03063] [SWS_SwCluC_CONSTR_03020] [SWS_SwCluC_CONSTR_03021] [SWS_SwCluC_CONSTR_03032]
[SRS_SwCluC_00105]	'1:n' External Trigger communication	[SWS_SwCluC_03046] [SWS_SwCluC_03048] [SWS_SwCluC_03049] [SWS_SwCluC_03050] [SWS_SwCluC_03051] [SWS_SwCluC_03052] [SWS_SwCluC_03055] [SWS_SwCluC_03056] [SWS_SwCluC_03058] [SWS_SwCluC_03060] [SWS_SwCluC_CONSTR_03053] [SWS_SwCluC_CONSTR_03054] [SWS_SwCluC_CONSTR_03059]
[SRS_SwCluC_00106]	'1:n' Parameter Communication	[SWS_SwCluC_03006] [SWS_SwCluC_03007] [SWS_SwCluC_03008] [SWS_SwCluC_03009] [SWS_SwCluC_03010] [SWS_SwCluC_03011] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00107]	Support unspecific preemption scenarios	[SWS_SwCluC_03084] [SWS_SwCluC_03085]
[SRS_SwCluC_00108]	Prevent from writing directly to memory of other Software Clusters	[SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_CONSTR_03404]
[SRS_SwCluC_00201]	Standardized AUTOSAR Interfaces for software components	[SWS_SwCluC_02002] [SWS_SwCluC_02003] [SWS_SwCluC_02121] [SWS_SwCluC_02301] [SWS_SwCluC_02302] [SWS_SwCluC_02303] [SWS_SwCluC_02309] [SWS_SwCluC_02310] [SWS_SwCluC_02311] [SWS_SwCluC_02316] [SWS_SwCluC_02317] [SWS_SwCluC_02318] [SWS_SwCluC_02323] [SWS_SwCluC_02324] [SWS_SwCluC_02330] [SWS_SwCluC_02334] [SWS_SwCluC_02703] [SWS_SwCluC_02723] [SWS_SwCluC_02724] [SWS_SwCluC_02725] [SWS_SwCluC_02726] [SWS_SwCluC_02727] [SWS_SwCluC_02740] [SWS_SwCluC_02741] [SWS_SwCluC_02742] [SWS_SwCluC_02743] [SWS_SwCluC_03204]
[SRS_SwCluC_00202]	Standardized Interfaces for local BSW modules	[SWS_SwCluC_02001] [SWS_SwCluC_02125] [SWS_SwCluC_02212] [SWS_SwCluC_02225] [SWS_SwCluC_02609] [SWS_SwCluC_02610] [SWS_SwCluC_02611] [SWS_SwCluC_02612] [SWS_SwCluC_02618] [SWS_SwCluC_02619] [SWS_SwCluC_02620] [SWS_SwCluC_02621] [SWS_SwCluC_02622] [SWS_SwCluC_02623] [SWS_SwCluC_02665] [SWS_SwCluC_02666] [SWS_SwCluC_02702] [SWS_SwCluC_02703] [SWS_SwCluC_02704] [SWS_SwCluC_02705] [SWS_SwCluC_02706] [SWS_SwCluC_02707] [SWS_SwCluC_02736] [SWS_SwCluC_02737] [SWS_SwCluC_02738] [SWS_SwCluC_02739] [SWS_SwCluC_03202] [SWS_SwCluC_03507] [SWS_SwCluC_03510] [SWS_SwCluC_03511] [SWS_SwCluC_03512] [SWS_SwCluC_03513] [SWS_SwCluC_03514] [SWS_SwCluC_03515] [SWS_SwCluC_03516] [SWS_SwCluC_03517] [SWS_SwCluC_10012]
[SRS_SwCluC_00203]	Id abstraction	[SWS_SwCluC_02000]
[SRS_SwCluC_00204]	Modular Software Cluster Connection	[SWS_SwCluC_02007] [SWS_SwCluC_02101] [SWS_SwCluC_02144] [SWS_SwCluC_02200] [SWS_SwCluC_02201] [SWS_SwCluC_02338] [SWS_SwCluC_02339] [SWS_SwCluC_02600] [SWS_SwCluC_02601] [SWS_SwCluC_02748] [SWS_SwCluC_02749] [SWS_SwCluC_03214] [SWS_SwCluC_03215]
[SRS_SwCluC_00205]	Safeguarding connections between Software Clusters	[SWS_SwCluC_00015] [SWS_SwCluC_00022] [SWS_SwCluC_00042] [SWS_SwCluC_00043] [SWS_SwCluC_00077] [SWS_SwCluC_03005] [SWS_SwCluC_03033] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03045] [SWS_SwCluC_03046] [SWS_SwCluC_03047]
[SRS_SwCluC_00206]	NV blocks in Application Software Cluster	[SWS_SwCluC_02101] [SWS_SwCluC_02102] [SWS_SwCluC_02103] [SWS_SwCluC_02104] [SWS_SwCluC_02105] [SWS_SwCluC_02106] [SWS_SwCluC_02107] [SWS_SwCluC_02108] [SWS_SwCluC_02109] [SWS_SwCluC_02110] [SWS_SwCluC_02111] [SWS_SwCluC_02112] [SWS_SwCluC_02113] [SWS_SwCluC_02114] [SWS_SwCluC_02115] [SWS_SwCluC_02116] [SWS_SwCluC_02117] [SWS_SwCluC_02118] [SWS_SwCluC_02119] [SWS_SwCluC_02120] [SWS_SwCluC_02121] [SWS_SwCluC_02122] [SWS_SwCluC_02123] [SWS_SwCluC_02124] [SWS_SwCluC_02125] [SWS_SwCluC_02127] [SWS_SwCluC_02128] [SWS_SwCluC_02129] [SWS_SwCluC_02132] [SWS_SwCluC_02133] [SWS_SwCluC_02136] [SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02142] [SWS_SwCluC_02143] [SWS_SwCluC_02144] [SWS_SwCluC_02147] [SWS_SwCluC_02149] [SWS_SwCluC_02150] [SWS_SwCluC_02151] [SWS_SwCluC_02152] [SWS_SwCluC_02153] [SWS_SwCluC_02154] [SWS_SwCluC_02155] [SWS_SwCluC_02156] [SWS_SwCluC_02157] [SWS_SwCluC_02161] [SWS_SwCluC_02162] [SWS_SwCluC_02163] [SWS_SwCluC_02164] [SWS_SwCluC_CONSTR_02130] [SWS_SwCluC_CONSTR_02131] [SWS_SwCluC_CONSTR_02134] [SWS_SwCluC_CONSTR_02135] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_SwCluC_CONSTR_02141] [SWS_SwCluC_CONSTR_02145] [SWS_SwCluC_CONSTR_02146] [SWS_SwCluC_CONSTR_02148] [SWS_SwCluC_CONSTR_02158] [SWS_SwCluC_CONSTR_02159] [SWS_SwCluC_CONSTR_02160] [SWS_SwCluC_CONSTR_02165]
[SRS_SwCluC_00208]	Dcm contribution by Application Software Cluster	[SWS_SwCluC_02300] [SWS_SwCluC_02301] [SWS_SwCluC_02302] [SWS_SwCluC_02303] [SWS_SwCluC_02304] [SWS_SwCluC_02305] [SWS_SwCluC_02306] [SWS_SwCluC_02307] [SWS_SwCluC_02308] [SWS_SwCluC_02309] [SWS_SwCluC_02310] [SWS_SwCluC_02311] [SWS_SwCluC_02312] [SWS_SwCluC_02313] [SWS_SwCluC_02314] [SWS_SwCluC_02315] [SWS_SwCluC_02316] [SWS_SwCluC_02317] [SWS_SwCluC_02318] [SWS_SwCluC_02319] [SWS_SwCluC_02320] [SWS_SwCluC_02321] [SWS_SwCluC_02322] [SWS_SwCluC_02323] [SWS_SwCluC_02324] [SWS_SwCluC_02325] [SWS_SwCluC_02330] [SWS_SwCluC_02331] [SWS_SwCluC_02332] [SWS_SwCluC_02333] [SWS_SwCluC_02334] [SWS_SwCluC_02335] [SWS_SwCluC_02336] [SWS_SwCluC_02337] [SWS_SwCluC_02338] [SWS_SwCluC_02339] [SWS_SwCluC_02341] [SWS_SwCluC_02342] [SWS_SwCluC_02350] [SWS_SwCluC_02351] [SWS_SwCluC_02352] [SWS_SwCluC_02353] [SWS_SwCluC_02354] [SWS_SwCluC_02355] [SWS_SwCluC_02356] [SWS_SwCluC_02357] [SWS_SwCluC_02358] [SWS_SwCluC_02359] [SWS_SwCluC_02360] [SWS_SwCluC_02361] [SWS_SwCluC_02362] [SWS_SwCluC_02363] [SWS_SwCluC_02364] [SWS_SwCluC_02365] [SWS_SwCluC_02366] [SWS_SwCluC_02367] [SWS_SwCluC_02368] [SWS_SwCluC_02369] [SWS_SwCluC_02370] [SWS_SwCluC_02371] [SWS_SwCluC_02372] [SWS_SwCluC_02373] [SWS_SwCluC_02374] [SWS_SwCluC_02375] [SWS_SwCluC_02376] [SWS_SwCluC_02380] [SWS_SwCluC_02381] [SWS_SwCluC_02382] [SWS_SwCluC_02383] [SWS_SwCluC_02384] [SWS_SwCluC_02385] [SWS_SwCluC_02386] [SWS_SwCluC_02387] [SWS_SwCluC_02388] [SWS_SwCluC_02400] [SWS_SwCluC_02401] [SWS_SwCluC_02402] [SWS_SwCluC_02403] [SWS_SwCluC_02404] [SWS_SwCluC_02405] [SWS_SwCluC_02406] [SWS_SwCluC_02407] [SWS_SwCluC_02408] [SWS_SwCluC_02409] [SWS_SwCluC_02410] [SWS_SwCluC_02411] [SWS_SwCluC_02412] [SWS_SwCluC_02413] [SWS_SwCluC_02414] [SWS_SwCluC_02415] [SWS_SwCluC_02416] [SWS_SwCluC_02417] [SWS_SwCluC_02418] [SWS_SwCluC_02419] [SWS_SwCluC_02420] [SWS_SwCluC_02421] [SWS_SwCluC_02422] [SWS_SwCluC_02423] [SWS_SwCluC_02424] [SWS_SwCluC_02425] [SWS_SwCluC_CONSTR_02427] [SWS_SwCluC_CONSTR_02428] [SWS_SwCluC_CONSTR_02429] [SWS_SwCluC_CONSTR_02752]
[SRS_SwCluC_00209]	Diagnostic Monitors in Application Software Cluster	[SWS_SwCluC_02700] [SWS_SwCluC_02701] [SWS_SwCluC_02702] [SWS_SwCluC_02703] [SWS_SwCluC_02704] [SWS_SwCluC_02705] [SWS_SwCluC_02706] [SWS_SwCluC_02707] [SWS_SwCluC_02708] [SWS_SwCluC_02709] [SWS_SwCluC_02710] [SWS_SwCluC_02711] [SWS_SwCluC_02712] [SWS_SwCluC_02713] [SWS_SwCluC_02714] [SWS_SwCluC_02715] [SWS_SwCluC_02716] [SWS_SwCluC_02717] [SWS_SwCluC_02718] [SWS_SwCluC_02719] [SWS_SwCluC_02723] [SWS_SwCluC_02724] [SWS_SwCluC_02725] [SWS_SwCluC_02726] [SWS_SwCluC_02727] [SWS_SwCluC_02728] [SWS_SwCluC_02729] [SWS_SwCluC_02730] [SWS_SwCluC_02731] [SWS_SwCluC_02732] [SWS_SwCluC_02733] [SWS_SwCluC_02734] [SWS_SwCluC_02735] [SWS_SwCluC_02736] [SWS_SwCluC_02737] [SWS_SwCluC_02738] [SWS_SwCluC_02739] [SWS_SwCluC_02740] [SWS_SwCluC_02741] [SWS_SwCluC_02742] [SWS_SwCluC_02743] [SWS_SwCluC_02744] [SWS_SwCluC_02745] [SWS_SwCluC_02746] [SWS_SwCluC_02747] [SWS_SwCluC_02748] [SWS_SwCluC_02749] [SWS_SwCluC_02751] [SWS_SwCluC_CONSTR_02720] [SWS_SwCluC_CONSTR_02721]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00210]	Usage of Function Inhibitions in Application Software Cluster	[SWS_SwCluC_03200] [SWS_SwCluC_03201] [SWS_SwCluC_03202] [SWS_SwCluC_03203] [SWS_SwCluC_03204] [SWS_SwCluC_03205] [SWS_SwCluC_03206] [SWS_SwCluC_03207] [SWS_SwCluC_03208] [SWS_SwCluC_03209] [SWS_SwCluC_03210] [SWS_SwCluC_03211] [SWS_SwCluC_03212] [SWS_SwCluC_03213] [SWS_SwCluC_03214] [SWS_SwCluC_03215] [SWS_SwCluC_03217] [SWS_SwCluC_03218] [SWS_SwCluC_CONSTR_03210] [SWS_SwCluC_CONSTR_03211]
[SRS_SwCluC_00211]	Com Invocation from Application Software Cluster	[SWS_SwCluC_02600] [SWS_SwCluC_02601] [SWS_SwCluC_02602] [SWS_SwCluC_02603] [SWS_SwCluC_02604] [SWS_SwCluC_02605] [SWS_SwCluC_02606] [SWS_SwCluC_02607] [SWS_SwCluC_02608] [SWS_SwCluC_02609] [SWS_SwCluC_02610] [SWS_SwCluC_02611] [SWS_SwCluC_02612] [SWS_SwCluC_02613] [SWS_SwCluC_02614] [SWS_SwCluC_02618] [SWS_SwCluC_02619] [SWS_SwCluC_02620] [SWS_SwCluC_02621] [SWS_SwCluC_02622] [SWS_SwCluC_02623] [SWS_SwCluC_02624] [SWS_SwCluC_02625] [SWS_SwCluC_02626] [SWS_SwCluC_02627] [SWS_SwCluC_02628] [SWS_SwCluC_02629] [SWS_SwCluC_02630] [SWS_SwCluC_02631] [SWS_SwCluC_02632] [SWS_SwCluC_02633] [SWS_SwCluC_02634] [SWS_SwCluC_02636] [SWS_SwCluC_02637] [SWS_SwCluC_02638] [SWS_SwCluC_02639] [SWS_SwCluC_02640] [SWS_SwCluC_02642] [SWS_SwCluC_02643] [SWS_SwCluC_02644] [SWS_SwCluC_02645] [SWS_SwCluC_02646] [SWS_SwCluC_02647] [SWS_SwCluC_02648] [SWS_SwCluC_02649] [SWS_SwCluC_02651] [SWS_SwCluC_02652] [SWS_SwCluC_02661] [SWS_SwCluC_02662] [SWS_SwCluC_02663] [SWS_SwCluC_02665] [SWS_SwCluC_02666] [SWS_SwCluC_02669] [SWS_SwCluC_02670] [SWS_SwCluC_03502] [SWS_SwCluC_03506] [SWS_SwCluC_03507] [SWS_SwCluC_03508] [SWS_SwCluC_03510] [SWS_SwCluC_03511] [SWS_SwCluC_03512] [SWS_SwCluC_03513] [SWS_SwCluC_03514] [SWS_SwCluC_03515] [SWS_SwCluC_03516] [SWS_SwCluC_03517] [SWS_SwCluC_03519] [SWS_SwCluC_03521] [SWS_SwCluC_03522] [SWS_SwCluC_03523] [SWS_SwCluC_03524] [SWS_SwCluC_03525] [SWS_SwCluC_03526] [SWS_SwCluC_03527] [SWS_SwCluC_03528] [SWS_SwCluC_03529] [SWS_SwCluC_03530] [SWS_SwCluC_03531] [SWS_SwCluC_03532] [SWS_SwCluC_03533] [SWS_SwCluC_03534] [SWS_SwCluC_03535] [SWS_SwCluC_03536] [SWS_SwCluC_03537] [SWS_SwCluC_03538] [SWS_SwCluC_03539] [SWS_SwCluC_03540] [SWS_SwCluC_03541] [SWS_SwCluC_03542] [SWS_SwCluC_03543] [SWS_SwCluC_03544] [SWS_SwCluC_03548] [SWS_SwCluC_03549] [SWS_SwCluC_03550] [SWS_SwCluC_03551] [SWS_SwCluC_03552] [SWS_SwCluC_03554] [SWS_SwCluC_10012] [SWS_SwCluC_13002] [SWS_SwCluC_13003] [SWS_SwCluC_13004] [SWS_SwCluC_13005] [SWS_SwCluC_13006] [SWS_SwCluC_13007] [SWS_SwCluC_91004] [SWS_SwCluC_91005] [SWS_SwCluC_91006] [SWS_SwCluC_91007] [SWS_SwCluC_91008] [SWS_SwCluC_91009] [SWS_SwCluC_91010] [SWS_SwCluC_91011] [SWS_SwCluC_CONSTR_02653] [SWS_SwCluC_CONSTR_02654] [SWS_SwCluC_CONSTR_02655] [SWS_SwCluC_CONSTR_02656] [SWS_SwCluC_CONSTR_02657] [SWS_SwCluC_CONSTR_02658] [SWS_SwCluC_CONSTR_02659] [SWS_SwCluC_CONSTR_02660] [SWS_SwCluC_CONSTR_02664] [SWS_SwCluC_CONSTR_02667] [SWS_SwCluC_CONSTR_03546] [SWS_SwCluC_CONSTR_03555] [SWS_SwCluC_CONSTR_03556] [SWS_SwCluC_CONSTR_03557] [SWS_SwCluC_CONSTR_03558] [SWS_SwCluC_CONSTR_03559] [SWS_SwCluC_CONSTR_03560] [SWS_SwCluC_CONSTR_03561] [SWS_SwCluC_CONSTR_03562]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00212]	Post Build connection between Software Clusters	[SWS_SwCluC_00048] [SWS_SwCluC_00049] [SWS_SwCluC_00050] [SWS_SwCluC_00051] [SWS_SwCluC_00052] [SWS_SwCluC_00053] [SWS_SwCluC_00075] [SWS_SwCluC_00076] [SWS_SwCluC_00077] [SWS_SwCluC_00086] [SWS_SwCluC_02008] [SWS_SwCluC_02009] [SWS_SwCluC_02390] [SWS_SwCluC_02391] [SWS_SwCluC_02392] [SWS_SwCluC_02393] [SWS_SwCluC_02394] [SWS_SwCluC_02395] [SWS_SwCluC_02396] [SWS_SwCluC_02397] [SWS_SwCluC_02646] [SWS_SwCluC_02647] [SWS_SwCluC_02662] [SWS_SwCluC_02663] [SWS_SwCluC_02744] [SWS_SwCluC_02745] [SWS_SwCluC_02746] [SWS_SwCluC_02747] [SWS_SwCluC_03003] [SWS_SwCluC_03012] [SWS_SwCluC_03013] [SWS_SwCluC_03142] [SWS_SwCluC_03210] [SWS_SwCluC_03211] [SWS_SwCluC_03212] [SWS_SwCluC_03213] [SWS_SwCluC_03544] [SWS_SwCluC_03551] [SWS_SwCluC_CONSTR_00078] [SWS_SwCluC_CONSTR_00096] [SWS_SwCluC_CONSTR_03063]
[SRS_SwCluC_00213]	Support unconnected interfaces from other Software Clusters	[SWS_SwCluC_02004] [SWS_SwCluC_02107] [SWS_SwCluC_02108] [SWS_SwCluC_02113] [SWS_SwCluC_02114] [SWS_SwCluC_02161] [SWS_SwCluC_02162] [SWS_SwCluC_02163] [SWS_SwCluC_02164] [SWS_SwCluC_02266] [SWS_SwCluC_02380] [SWS_SwCluC_02381] [SWS_SwCluC_02382] [SWS_SwCluC_02383] [SWS_SwCluC_02384] [SWS_SwCluC_02385] [SWS_SwCluC_02386] [SWS_SwCluC_02387] [SWS_SwCluC_02388] [SWS_SwCluC_02402] [SWS_SwCluC_02405] [SWS_SwCluC_02408] [SWS_SwCluC_02411] [SWS_SwCluC_02414] [SWS_SwCluC_02417] [SWS_SwCluC_02420] [SWS_SwCluC_02423] [SWS_SwCluC_02612] [SWS_SwCluC_02613] [SWS_SwCluC_02614] [SWS_SwCluC_02634] [SWS_SwCluC_02710] [SWS_SwCluC_02711] [SWS_SwCluC_02712] [SWS_SwCluC_02713] [SWS_SwCluC_02714] [SWS_SwCluC_02715] [SWS_SwCluC_02716] [SWS_SwCluC_02717] [SWS_SwCluC_02718] [SWS_SwCluC_02734] [SWS_SwCluC_02735] [SWS_SwCluC_03010] [SWS_SwCluC_03107] [SWS_SwCluC_03112] [SWS_SwCluC_03130] [SWS_SwCluC_03131] [SWS_SwCluC_03137] [SWS_SwCluC_03143] [SWS_SwCluC_03164] [SWS_SwCluC_03167] [SWS_SwCluC_03205] [SWS_SwCluC_03206] [SWS_SwCluC_03405] [SWS_SwCluC_03508] [SWS_SwCluC_03530] [SWS_SwCluC_03531] [SWS_SwCluC_03532] [SWS_SwCluC_CONSTR_02134] [SWS_SwCluC_CONSTR_02165]
[SRS_SwCluC_00214]	Software Cluster local RTE	[SWS_SwCluC_02202] [SWS_SwCluC_02204] [SWS_SwCluC_02205] [SWS_SwCluC_02206] [SWS_SwCluC_02210] [SWS_SwCluC_02211] [SWS_SwCluC_02212] [SWS_SwCluC_02213] [SWS_SwCluC_02214] [SWS_SwCluC_02215] [SWS_SwCluC_02216] [SWS_SwCluC_02217] [SWS_SwCluC_02219] [SWS_SwCluC_02220] [SWS_SwCluC_02221] [SWS_SwCluC_02222] [SWS_SwCluC_02223] [SWS_SwCluC_02224] [SWS_SwCluC_02225] [SWS_SwCluC_02226] [SWS_SwCluC_02227] [SWS_SwCluC_02229] [SWS_SwCluC_02230] [SWS_SwCluC_02250] [SWS_SwCluC_02251] [SWS_SwCluC_02252] [SWS_SwCluC_02253] [SWS_SwCluC_02254] [SWS_SwCluC_02255] [SWS_SwCluC_02256] [SWS_SwCluC_02259] [SWS_SwCluC_02262] [SWS_SwCluC_02263] [SWS_SwCluC_02264] [SWS_SwCluC_02265] [SWS_SwCluC_02271] [SWS_SwCluC_02272] [SWS_SwCluC_02273] [SWS_SwCluC_02274] [SWS_SwCluC_02275] [SWS_SwCluC_02276] [SWS_SwCluC_02277] [SWS_SwCluC_02278] [SWS_SwCluC_02279] [SWS_SwCluC_02280] [SWS_SwCluC_02284] [SWS_SwCluC_02285] [SWS_SwCluC_CONSTR_02203] [SWS_SwCluC_CONSTR_02231] [SWS_SwCluC_CONSTR_02232] [SWS_SwCluC_CONSTR_02233] [SWS_SwCluC_CONSTR_02234] [SWS_SwCluC_CONSTR_02235] [SWS_SwCluC_CONSTR_02237] [SWS_SwCluC_CONSTR_02281] [SWS_SwCluC_CONSTR_02282] [SWS_SwCluC_CONSTR_02283]
[SRS_SwCluC_00300]	A2L Generation Support	[SWS_SwCluC_03071] [SWS_SwCluC_03072] [SWS_SwCluC_03073] [SWS_SwCluC_03074] [SWS_SwCluC_03075] [SWS_SwCluC_03076] [SWS_SwCluC_03077]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Binary Manifest

7.1.1 Overview

With the concept of Software Clusters, the overall software of an AUTOSAR Classic Platform Architecture can be split into smaller units. In such a clustered AUTOSAR Classic Platform Architecture a resource is used to describes any capability which

- is needed to operate the software in Software Clusters

  AND

- which is provided by one Software Cluster for another Software Clusters

Please note as well document [13] with its description about CpSoftwareClusterResources.

Each Software Cluster is an independent build unit, and the result of the cluster-specific build processes are the Binary Objects. The Binary Manifests provide the means to connect the Binary Objects that are deployed on the same machine. Hence, the Binary Manifest is the well-defined interface of the Software Cluster's Binary Object.

The Binary Manifest provides any information, which is required to access a resource inside a Software Cluster, and to connect provided and required resources of Software Clusters. A resource, in this context, can be anything that is required to operate the software. For example, a sender receiver interface of the Software Cluster or a NV block. An obvious property of such a resource is, whether it is provided or required by a Software Cluster.

The Binary Manifest has the following core characteristics:

- The Binary Manifest gets created during the build of the Software Cluster, since it has to store information which might be build dependent (e.g. data and function addresses, ID values to use BSW APIs, attribute values, hashes)
- The Binary Manifest provides a C interface towards Software Cluster's implementation. This supports an abstraction in the other functional blocks of the Software Cluster Connection from the Binary Manifest's table implementation. The C interface is also accessible by implementations of CDDs requiring the usage of the Binary Manifest.
- The Binary Manifest defines an ECU-C interface to Software Cluster's build tools. This supports an abstract usage of the Binary Manifest by other functional blocks of the Software Cluster Connection, as well as the usage of the Binary Manifest by CDDs.

- The `Binary Manifest` format can be easily interpreted by the target `machine`. Note: In contrast to concepts for other domains (e.g. Java, Android), in this specification it is not the goal to provide a textual manifest (XML, JSON).
- An unique identifier per resource is used for the connection process. These are explicitly assigned since hash numbers are not suitable.
- The `Binary Manifest` shall have guarding information, to ensure that only compatible interfaces are getting connected. These guarding values consist of a hash over certain interface properties (see 7.3.2.2 and 7.5).

7.1.2 Logical structure of a Binary Manifest

The Figure 7.1 provides an overview of the conceptual elements of the `Binary Manifest` necessary for a fictional example resource.

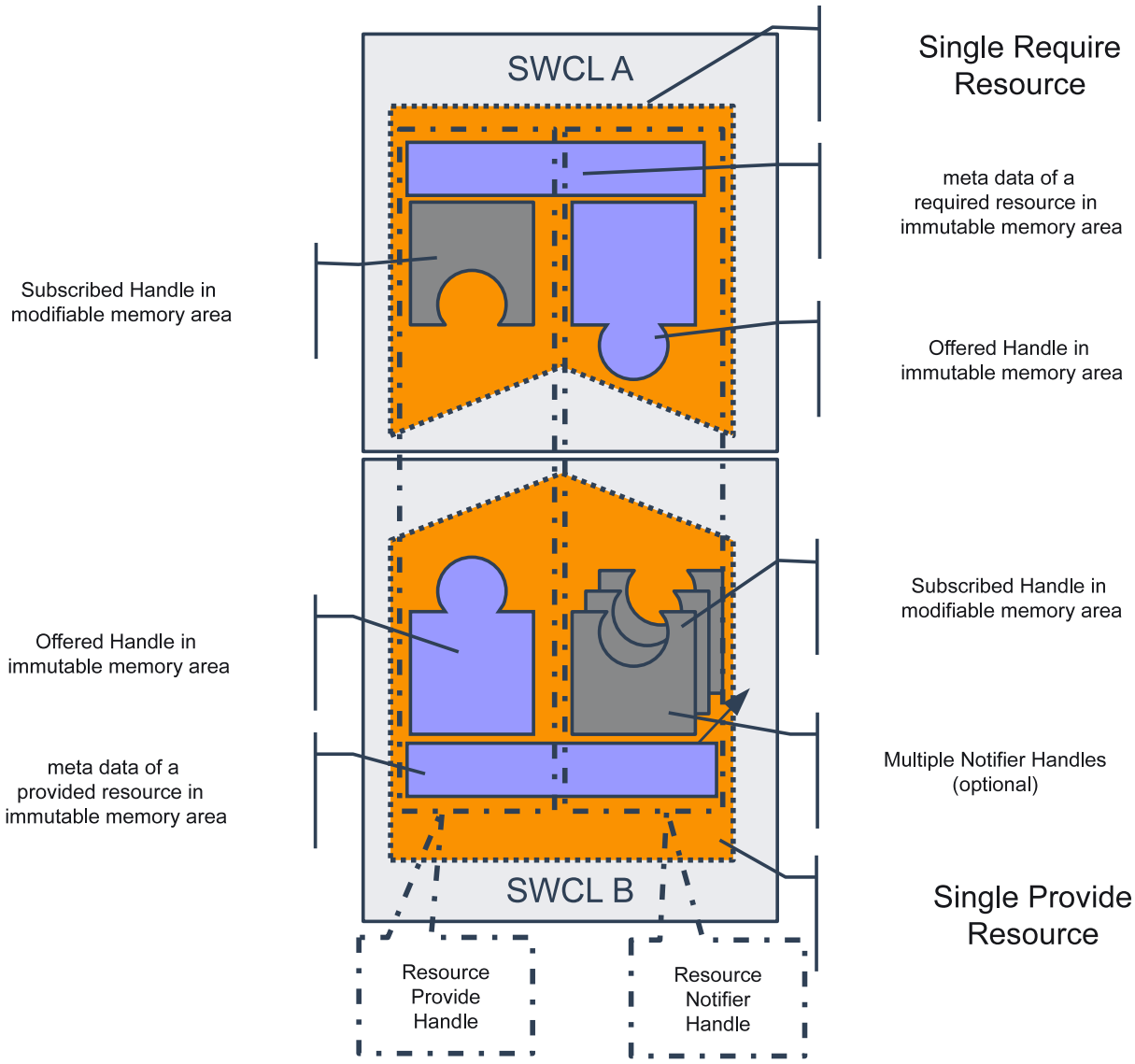

Figure 7.1: Logical structure of a Binary Manifest

The Binary Manifest provides the ability to connect required and provided resources of Software Clusters. Each resource is qualified by a set of meta data, e.g.

- the unique identifier of the resource
- the type of the resource (e.g. a sender receiver communication or connection of NV blocks)
- the nature, whether a resource is required or provided by this Software Cluster
- in case of required resources, whether it is mandatory for the operation of the Software Cluster
- guarding information (e.g. hash values) to ensure that only compatible resources are getting connected
- the number of stored handles, and where those handles can be found in the the according tables

Each type of resource is qualified by the number of handles, their purpose for the connection, and the order in which they are placed in the Binary Manifest. The handles hold the information, which the Software Cluster requesting a resource needs to access it in the providing Software Cluster.

A single handle can be

- a data pointer
- a function pointer
- a value

In some cases, a connection of a resource may require the exchange of handles in both directions. Consistently the Binary Manifest distinguishes between two types of handles on the logical level:

- Provide Handle to publish a handle from Resource Provider to Resource Requester
- Notifier Handle to publish a handle from Resource Requester to Resource Provider

An example for a Provide Handle would be a propagated API function to call an AUTOSAR Service from an Application Software Cluster.

An example for a Notifier Handle would be a callback notification from a connected AUTOSAR Service.

In addition, it is supported that a Resource Provider gets connected to multiple Resource Requesters even if it utilizes Notifier Handles. But in this case for each potential Resource Requester a own set of Notifier Handles needs to be reserved for each Resource Requester. Hence a Notifier Handle Set holds all the Notifier Handles for one Resource Requester.

The Binary Manifest is composed of the following parts:

- the Binary Manifest Header. It is the central entry point, provides administrative data and references to the other parts of the manifest.
- the Interface Descriptor Table. It contains one row of meta information per Resource Entry (required or provided interface) - unique ID, properties to operate the resource at run-time, hash values about interface characteristics and semantics.
- the Offered Interface. It contains all handles that are offered to other clusters - as explained above, either because a resource is provided, or because a required resource also needs a handle in the other direction.
- the Subscribed Interface. Vice versa, it contains all handles that this Software Cluster holds that were offered by other Software Clusters
- checksums and markers (Immutable Tables Checksum, Subscribed Interface Validity Marker)

Some parts of the Binary Manifest are fixed when building a Software Cluster, called immutable, while others have to be changed in the connection process, called modifiable:

- immutable
  - the Binary Manifest Header
  - the Interface Descriptor Table
  - offered handles
  - the default value for subscribed handles, to support operation in the unconnected state
- modifiable

  To establish a connection, this memory area has to store the handles from other SWCLs. Additionally, information might be stored to identify whether the resource is connected at all (only defaults are visible) and to which SWCL.

7.1.3 Mapping between Logical structure and Configuration structure

The logical structure shown in Figure 7.1 is also reflected in the configuration structure of the Binary Manifest.

The set of handles, relevant to connect one type of resource, is defined in the SwCluCBManifResourceType. A Resource Type is characterized by a defined set of Provide Handles and Notifier Handles, with specific types in a well defined order.

Handles can optionally exist. This is useful for resources which do not always need the full set of handles in all feature configurations. For instance, sender receiver communication works with or without sendIndication. Spending always a new resource type id (SwCluCBManifResourceTypeId) is not appropriate due to the limited numberspace! But the Resource Provider and Resource Requester have to have the same mutual understanding of the optionality conditions and both have to see a matching configuration to determine the identical handle configuration for a particular CpSoftwareClusterResource.

This is the prerequisite to enable the connection between Software Clusters.

Optional handles are always added after the non-optional ones!

In addition it is highly recommended that the configuration values resulting in different handle configurations are also considered for the guard value calculation.

Each required or provided resource of this Software Cluster corresponds to one Resource Entry in the Binary Manifest.

One Resource Entry corresponds to exactly one row in the Interface Descriptor Table, plus one or several handles in the Offered Interface and / or Subscribed Interface. Depending whether the resource is required or provided, Provide Handles and Notifier Handles are implemented in the Software Cluster or expected to be set by the other Software Cluster. Please note as well [SWS_SwCluC_00088].

Software Clusters providing a resource and requiring a resource use the identical definition of the Resource Type. The providing Software Cluster configures a SwCluCBManifProvideResourceEntry whereas the requiring Software Cluster configures a SwCluCBManifRequireResourceEntry. Depending on weather the handle is a Provide Handle or a Notifier Handle the parameters SwCluCBManifProvideSymbol or SwCluCBManifDefaultNotifierSymbol are used. See table 7.1.

Resource Type	Providing Software Cluster	Requiring Software Cluster
SwCluCBManifProvideHandle	SwCluCBManifProvideResourceEntry. SwCluCBManifProvideSymbol	SwCluCBManifRequireResourceEntry. SwCluCBManifDefaultProvideSymbol
SwCluCBManifNotifierHandle	SwCluCBManifProvideResourceEntry. SwCluCBManifDefaultNotifierSymbol	SwCluCBManifRequireResourceEntry. SwCluCBManifNotifierSymbol

Table 7.1: Provide Resource Entry and Require Resource Entry configuraton usage

Therefore, the configuration provides two distinct definitions for Provide Resource Entrys as SwCluCBManifProvideResourceEntry and Require Resource Entrys as SwCluCBManifRequireResourceEntry.

From a Binary Manifest user perspective, it is an important use case to iterate over uniform Resource Entrys by an index. For this purpose, the configuration provides the ability to group Resource Entrys of the same Resource Type in Resource Entry Groups. A Provide Resource Entry Group, being a group of Provide Resource Entrys, is represented as SwCluCBManifProvideResourceEntryGroup. Correspondingly, a Require Resource Entry Group, being a group of Require Resource Entrys, is represented as SwCluCBManifRequireResourceEntryGroup.

In this and the previous section, three pairs of terms have been introduced, for three different logical layers within the Binary Manifest:

1. provide and require resources
2. Provide Handles and Notifier Handles
3. offered and subscribed handles

A single connection (between provide and require resource, 1), consists of one or more logical channels (provide and notifier handles, 2), which are implemented in the binary manifest (offered and subscribed handles, 3).

An analogy might help to clarify these terms.

Consider a satellite receiver connected to a TV via a cable, which internally contains many wires. The satellite receiver provides 'video', the TV requests it (provide and require resource). To make a single connection between provide and require side, multiple separate data flows (provide and notifier handles) are required. Some transmit the video and audio signal from the provider to the requestor, while others transmit remote control signals in the opposite direction. Both the TV and the satellite receiver have a connector with many pins (provide and notifier handles).

For the 'left audio channel' (Provide Handle), a certain pin is used. On that pin, the satellite receiver sends a signal (an offered handle implements this Provide Handle), which the TV reads from the same pin (a subscribed handle implements this Provide Handle). For the 'remote control signal' (Notifier Handle) pin, the TV sends the signal (offered handle implements the Notifier Handles), while the satellite receiver reads the signal (subscribed handle implements the Notifier Handles).

The possible combinations of the three layers are shown in [SWS_SwCluC_00088].

7.1.4 Implementation structure of a Binary Manifest

Usually, in AUTOSAR the implementation of BSW modules is not standardized. But the various data tables of the Binary Manifest implement an interface to Software Cluster connector algorithms. Therefore, the implemented layout and semantic of the tables is standardized in this document. Furthermore, an abstract set of requirements for such connector algorithms is defined in section 7.1.5.

The Figure 7.2 provides an overview about the individual tables of the `Binary Manifest`, and their main relationships.

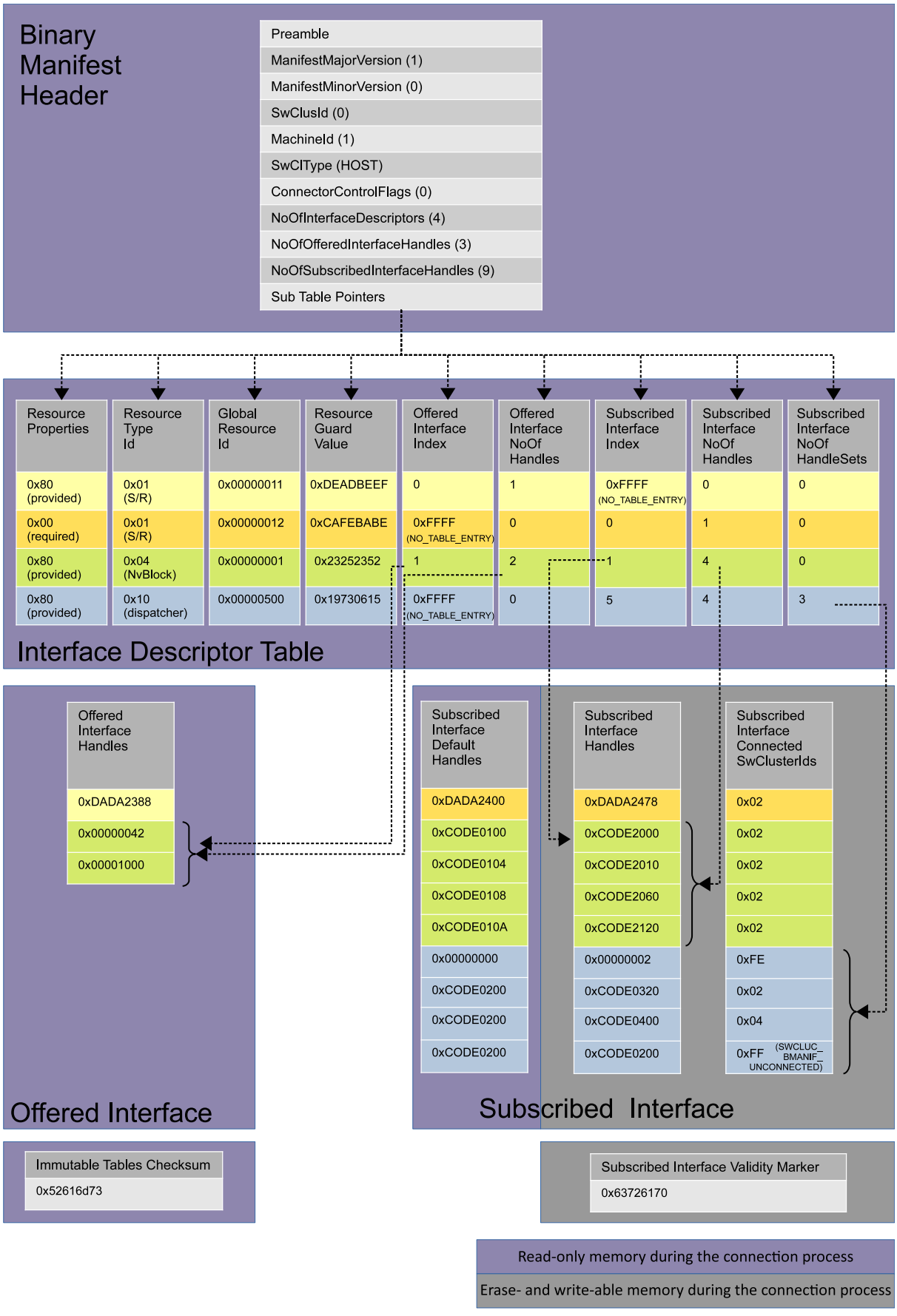

Figure 7.2: Implementation structure of a Binary Manifest

7.1.4.1 Implementation Overview

Each Software Cluster's Binary Manifest has a central entry point, called the Binary Manifest Header. It contains the AUTOSAR standardized meta data of a Software Cluster, and pointers to the sub tables (and their columns, as explained below) of the Binary Manifest. It is specified in 7.1.4.3.

Each row in the Interface Descriptor Table stores the properties of a single resource. For instance, the global resource Id. To avoid alignment gaps and padding bytes, which could occur if multiple elements are put in a structure, the table is not stored as an array of such a structure. Instead, the individual properties are organized in separate arrays. The individual arrays of the Interface Descriptor Table are required to have exactly the same number of elements and the identical order. In other words, these arrays can be seen as the columns of the Interface Descriptor Table and the identical index denotes the table row. Each row describes one resource. It is specified in 7.1.4.4.

Each of these resources can now use a dedicated number of handles in the offered and / or subscribed handle tables. The indication of the associated handles is done with the start index for the interface table(s) and the number of handles in the interface table(s). Since the number and placement of offered and subscribed handles can differ, a separate 'index' and a separate 'number of handles' for offered and subscribed handle tables exist. A special NO_TABLE_ENTRY value in the index indicates that no handle exists in that table. In this case, the number of handles is set to zero.

For example, in figure 7.2, the resource with id 0x11 (marked yellow) has 1 offered handle at index 0 and no subscribed handle (indicated by NO_TABLE_ENTRY). The resource with id 0x1 (marked green), has two handles in the offered table (starting at index 1) and four handles in the subscribed table (also starting at index 1).

The Offered Interface consists of one array - the Offered Interface Handle Column - to store the fixed handles, which are offered for the connection process (for example, in the case of a send port, the addresses of the data buffer related to the port that this software cluster provides). It is specified in 7.1.4.5

The Subscribed Interface consists out of three arrays:

- Subscribed Interface Default Handle Column offers default values for the subscribed handles, which are used in case the connection process does not have a connection partner in another Software Cluster.
- Subscribed Interface Handle Column holds the handles modified by the connection process. Those values are taken during the connection process from the Offered Interface Handle Column of another Software Cluster.
- The Subscribed Interface Connected SwClusterId Column holds the Software Cluster Id, from which the handle values are taken.

It is specified in 7.1.4.6.

Please note that the Subscribed Interface Handle Column and Subscribed Interface Connected SwClusterId Column need to be located in a memory area that can be erased and re-written during the connection process. In contrast, the Subscribed Interface Default Handle Column (like all the other Binary Manifest tables) needs to be located in a memory area which is read-only during the connection process. This split ensures that the connection process can be re-started at any point of time - even if the Subscribed Interface Handle Column and Subscribed Interface Connected SwClusterId Column is already erased and not yet re-written.

7.1.4.2 Multiple Notifier Sets Introduction

To support 1:n connections that require individual notifications of the requesters, the Binary Manifest supports the storage of multiple Notifier Handle Sets. In this case, the SubscribedInterfaceNoOfHandleSets field describes how many handles sets of other Software Clusters can be connected. Please note that the value 0 means that only a single connection is supported. 1..n means multiple connections are possible. If this is selected, the first handle entry is used to store the actual number of connected Software Clusters - similar to a dynamic length array. With this handling, it possible to preserve the MULTIPLE_NOTIFIER_SETS semantic for a Binary Manifest user, even if only at most one connection is supported.

It is also possible to connect interfaces that require more than one handle per connection. The number of handles per connection is not stored in the Binary Manifest, but if it is required, it can be calculated as (SubscribedInterfaceNoOfHandles -1) / SubscribedInterfaceNoOfHandleSets.

As an example, in figure 7.2, the resource with id 0x500 (marked blue) has 3 handle sets (the entry in the column SubscribedInterfaceNoOfHandleSets is set to 3). So at most three connections can be made to this resource. In this example, each of the three handle set consists of one handle, and since one more row is used for meta information, SubscribedInterfaceNoOfHandles is set to 3*1 + 1 = 4.

As explained above, the number of established connections is stored in the first handle entry. In this case, the entry is set to 0x2, so two connections are used and one is unused. The value 0xFE in SubscribedInterfaceConnectedClusterIds additionally indicates that this row does not contain a handle. The next two rows each show the target address of the handle and the cluster id. Since the last connection is currently unused, in the last row the handle value is set to the default value, and SubscribedInterfaceConnectedClusterIds is set to SWCLUC_BMANIF_UNCONNECTED (0xFF).

More in depth information about multiple Notifier Handle Sets can be found in 7.1.5.4. As part of that chapter, table 7.2 also shows an example with four sets, where each set consists of two handles.

7.1.4.3 Binary Manifest Header

[SWS_SwCluC_00001]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Binary Manifest Header

```
const SwCluC_BManif_HeaderType SwCluC_BManif_Header =
  { <initialization> };
```

⌋

[SWS_SwCluC_00002]
Upstream requirements: SRS_SwCluC_00006

⌈The header file SwCluC_BManif.h shall export the declaration of the Binary Manifest Header

```
extern const SwCluC_BManif_HeaderType SwCluC_BManif_Header;
```

⌋

[SWS_SwCluC_00003]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element Preamble of the Binary Manifest Header shall be set to the value 0x41524350464C4558.⌋

Note: The purpose of the Preamble is to serve as an obvious marker of the Binary Manifest's begin in memory. In addition, the value is chosen so that byte and word order can be detected, when the Binary Manifest is read in a byte stream or from a Binary Object file.

[SWS_SwCluC_00004]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element ManifestMajorVersion of the Binary Manifest Header shall be set to the value 0x01.⌋

[SWS_SwCluC_00005]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element ManifestMinorVersion of the Binary Manifest Header shall be set to the value 0x01.⌋

[SWS_SwCluC_00006]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element SwClusterId of the Binary Manifest Header shall be set to the value of the configuration parameter SwCluCSoftwareClusterId of the selected SwCluCDefinition.⌋

[SWS_SwCluC_00007]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element MachineId of the Binary Manifest Header shall be set to the value of the configuration parameter SwCluCMachineId of the selected SwCluCDefinition.⌋

[SWS_SwCluC_00008]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014, SRS_SwCluC_00005

⌈The element SwClusterType of the Binary Manifest Header shall be set according to the value of the configuration parameter SwCluCSoftwareClusterType of the selected SwCluCDefinition with the following encoding:

- HOST_SW_CLUSTER : 0x00
- APPLICATION_SW_CLUSTER : 0x01
- SUBSTITUTION_SW_CLUSTER : 0xFF

⌋

[SWS_SwCluC_00009]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The bit SWCLUC_BMANIF_DISABLE_ON_ECU_CONNECTION in the element ConnectorControlFlags of the Binary Manifest Header shall be set if the SwCluCBManifOnBoardConnectorControl is set to DISABLE_ON_ECU_CONNECTOR. Otherwise, the bit is not set (value 0).⌋

Note: It is not specified, if it is possible to reflash a cluster with changed addresses, how a system would detect this, and how it would behave in this case. This might change in future releases of this specification.

[SWS_SwCluC_00010]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element NoOfInterfaceDescriptors of the Binary Manifest Header shall be set to the number of SwCluCBManifProvideResourceEntry + number of SwCluCBManifRequireResourceEntry containers in the configuration.⌋

[SWS_SwCluC_00011]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element `NoOfOfferedInterfaceHandles` of the `Binary Manifest Header` shall be set to the number of offered handles.⌋

[SWS_SwCluC_00012]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element `NoOfSubscribedInterfaceHandles` of the `Binary Manifest Header` shall be set to the number of handles in the `Subscribed Interface`.⌋

[SWS_SwCluC_00070]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element `ImmutableTablesChecksumPtr` of the `Binary Manifest Header` shall reference the `Immutable Tables Checksum`.⌋

[SWS_SwCluC_00072]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014

⌈The element `SubscribedInterfaceValidityMarkerPtr` of the `Binary Manifest Header` shall reference the `Subscribed Interface Validity Marker`.⌋

[SWS_SwCluC_00042]
Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014, SRS_SwCluC_00205

⌈The elements

- `ResourcePropertiesDescriptorColumnPtr`
- `ResourceTypeDescriptorColumnPtr`
- `GlobalResourceIdDescriptorColumnPtr`
- `ResourceGuardValueDescriptorColumnPtr`
- `OfferedInterfaceIndexDescriptorColumnPtr`
- `OfferedInterfaceNoOfHandlesDescriptorColumnPtr`
- `SubscribedInterfaceIndexDescriptorColumnPtr`
- `SubscribedInterfaceNoOfHandlesDescriptorColumnPtr`
- `SubscribedInterfaceNoOfHandleSetsDescriptorColumnPtr`
- `OfferedInterfaceHandleColumnPtr`
- `SubscribedInterfaceHandleDefaultColumnPtr`

- SubscribedInterfaceHandleColumnPtr
- SubscribedInterfaceConnectedSwClusterIdColumnPtr

of the Binary Manifest Header shall reference the according array of the Interface Descriptor Table if the SwCluCBManifDescriptorTreatment is set to EMBED_DESCRIPTORS. Otherwise the elements are initialized to NULL_PTR.⌋

The order of the elements given by the data type SwCluC_BManif_HeaderType.

7.1.4.4 Interface Descriptor Table

[SWS_SwCluC_00015]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00002, SRS_SwCluC_00009, SRS_SwCluC_00010, SRS_SwCluC_00011, SRS_SwCluC_00205

⌈If the SwCluCBManifDescriptorTreatment is set to EMBED_DESCRIPTORS the Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Interface Descriptor Table composed out of the following column arrays:

```
1  /* descriptor table column for resource properties  */
2  const SwCluC_BManif_ResourcePropertiesType
       SwCluC_BManif_ResourcePropertiesDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
3
4  /* descriptor table column for resource type Ids  */
5  const SwCluC_BManif_ResourceTypeIdType
       SwCluC_BManif_ResourceTypeIdDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
6
7  /* descriptor table column for global resource Ids  */
8  const SwCluC_BManif_GlobalResourceIdType
       SwCluC_BManif_GlobalResourceIdDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
9
10 /* descriptor table column for guard values */
11 const SwCluC_BManif_ResourceGuardValueType
       SwCluC_BManif_ResourceGuardValueDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
12
13 /* descriptor table column for offered interface table index */
14 const SwCluC_BManif_TableIndexType
       SwCluC_BManif_OfferedInterfaceIndexDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
15
16 /* descriptor table column for number of handles in offered interface
       table*/
17 const SwCluC_BManif_HandleIndexType
       SwCluC_BManif_OfferedInterfaceNoOfHandlesDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
18
```

```
19 /* descriptor table column for subscribed interface table index */
20 const SwCluC_BManif_TableIndexType
       SwCluC_BManif_SubscribedInterfaceIndexDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
21
22 /* descriptor table column for number of handles in subscribed
       interface table*/
23 const SwCluC_BManif_HandleIndexType
       SwCluC_BManif_SubscribedInterfaceNoOfHandlesDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
24
25 /* descriptor table column for number of handle sets in subscribed
       interface table*/
26 const SwCluC_BManif_HandleIndexType
       SwCluC_BManif_SubscribedInterfaceNoOfHandleSetsDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
```

⌋

[SWS_SwCluC_00016]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈For each SwCluCBManifProvideResourceEntry and SwCluCBManifRequire-ResourceEntry in a Software Cluster's configuration, the Binary Manifest of the Software Cluster Connection shall provide one row in the Interface Descriptor Table.⌋

Note: This means that each array in the Interface Descriptor Table gets one element per row. In the below requirements, the term 'element of the X column' refers to one cell of the Interface Descriptor Table at a certain row and column. In this way, the content of each cell of the Interface Descriptor Table is specified.

[SWS_SwCluC_00017]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The rows in the Interface Descriptor Table shall be sorted in ascending order of resource type Ids and rows with equal resource type Ids shall be sorted in turn in ascending order of global resource Ids.⌋

[SWS_SwCluC_00018]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The bit SWCLUC_BMANIF_PROVIDED_RESOURCE in the element of the 'resource properties' column shall be set, if the row belongs to a SwCluCBManifProvide-ResourceEntry. Otherwise the bit SWCLUC_BMANIF_PROVIDED_RESOURCE shall not be set.⌋

[SWS_SwCluC_00019]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00009

⌈The bit SWCLUC_BMANIF_MANDATORY_RESOURCE in the element of the 'resource properties' column shall be set, if the row belongs to a SwCluCBManifRequire-

ResourceEntry where the SwCluCBManifIsMandatory is true. Otherwise the bit SWCLUC_BMANIF_MANDATORY_RESOURCE shall not be set.⌋

Please note that for SwCluCBManifProvideResourceEntrys the SWCLUC_BMANIF_MANDATORY_RESOURCE bit is always 0.

[SWS_SwCluC_00090]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The reserved bits 5 .. 0 in the element of the 'resource properties' column shall be set to 0⌋

[SWS_SwCluC_00020]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The element in the 'resource type Id' column shall be set to the SwCluCBManifResourceTypeId of the applicable SwCluCBManifResourceType.⌋

Note: The applicable SwCluCBManifResourceType is referenced by the owning SwCluCBManifProvideResourceEntryGroup / SwCluCBManifRequireResourceEntryGroup

[SWS_SwCluC_00021]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00002, SRS_SwCluC_00014

⌈The element in the 'global resource Id' column shall be set to the attribute value globalResourceId of the referenced CpSoftwareClusterResource as given via SwCluCBManifResourceRef. If the reference SwCluCBManifResourceRef is not set, the element in the global resource Id column shall be set to 0.⌋

[SWS_SwCluC_00022]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00010, SRS_SwCluC_00014, SRS_SwCluC_00205

⌈The element in the 'guard value' column shall be set to the SwCluCBManifResourceGuardValue of the SwCluCBManifProvideResourceEntry / SwCluCBManifRequireResourceEntry container in the configuration.⌋

[SWS_SwCluC_00023]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The element in the 'offered interface table index' column shall be set to the index of the Offered Interface where the first offered handle for this resource is allocated. In case the resource has no offered handle the value is set to NO_TABLE_ENTRY.⌋

[SWS_SwCluC_00024]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00003

⌈The element in the 'number of handles in offered interface table' column shall be set to number of offered handles for this resource. In case the resource has no offered handle, the value is set to 0.⌋

[SWS_SwCluC_00025]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The element in the 'subscribed interface table index' column shall be set to the index of the `Subscribed Interface` where the first subscribed handle for this resource is allocated. In case the resource has no subscribed handle the value is set to `NO_TABLE_ENTRY`.⌋

[SWS_SwCluC_00026]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00003

⌈The element in the 'number of handles in subscribed interface table' column shall be set to number of subscribed handles for this resource. In case the resource has no subscribed handle, the value is set to 0.⌋

[SWS_SwCluC_00027]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00004

⌈The element in the 'number of handle sets in subscribed interface table' column shall be set to the `SwCluCBManifMaxNumberOfNotifierSets` if the applicable `SwCluCBManifResourceType` has set `SwCluCBManifMultipleNotifierSupport` to `MULTIPLE_NOTIFIER_SETS`. Otherwise the value is set to 0.⌋

Depending whether a resource is provided or required by a `Software Cluster`, the `Provide Handles` and `Notifier Handles` need to be put either in the `Offered Interface` or in the `Subscribed Interface`. [SWS_SwCluC_00088] defines, how many of the handles for a `SwCluCBManifProvideResourceEntry` / `SwCluCBManifRequireResourceEntry` are created in the `Offered Interface` or `Subscribed Interface`.

[SWS_SwCluC_00088] Number of offered and subscribed handles, depending on the resource direction

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00004

⌈

	Provide Handle	Notifier Handle
Provided Resource	One handle in the `Offered Interface` per defined `SwCluCBManifProvideHandle` The handles are initialized with the `SwCluCBManifProvideSymbol`s in the given order.	**SINGLE_NOTIFIER_SET** One handle in the `Subscribed Interface` per defined `SwCluCBManifNotifierHandle` The handles are initialized with the `SwCluCBManifDefaultNotifierSymbol`s in the given order. The Software Cluster Ids are set to `SWCLUC_BMANIF_UNCONNECTED`
		MULTIPLE_NOTIFIER_SETS One handle in the `Subscribed Interface` per defined `SwCluCBManifNotifierHandle` multiplied by `SwCluCBManifMaxNumberOfNotifierSets` plus one The first handle is initialized to 0. The remaining handles are initialized with consecutive sets of `SwCluCBManifDefaultNotifierSymbol`s in the given order. The Software Cluster Ids are set to `SWCLUC_BMANIF_UNCONNECTED`.
Required Resource	One handle in the `Subscribed Interface` per defined `SwCluCBManifProvideHandle` The handles are initialized with the `SwCluCBManifDefaultProvideSymbol`s in the given order.	One handle in the `Offered Interface` per defined `SwCluCBManifNotifierHandle` The handles are initialized with the `SwCluCBManifNotifierSymbol`s in the given order.

⌋

7.1.4.5 Offered Interface Table

[SWS_SwCluC_00028]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011

⌈The `Binary Manifest` of the `Software Cluster Connection` shall provide exactly one instance of the `Offered Interface Handle Column`

```
const SwCluC_BManif_HandleType
    SwCluC_BManif_OfferedInterfaceHandleColumn[
    SWCLUC_BMANIF_NO_OF_OFFERED_HANDLES] = { <initialization> };
```

⌋

[SWS_SwCluC_00029]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013

⌈The `Binary Manifest` of the `Software Cluster Connection` shall allocate and initialize the number of handles in the `Offered Interface` according to [SWS_SwCluC_00088].⌋

7.1.4.6 Subscribed Interface Table

[SWS_SwCluC_00030]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00009

⌈The `Binary Manifest` of the `Software Cluster Connection` shall provide exactly one instance of the `Subscribed Interface Default Handle Column`

```
const SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceDefaultHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = { <initialization> };
```

⌋

[SWS_SwCluC_00031]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013

⌈The `Binary Manifest` of the `Software Cluster Connection` shall allocate and initialize the number of handles in the `Subscribed Interface Default Handle Column` according to [SWS_SwCluC_00088].⌋

[SWS_SwCluC_00032]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013

⌈The `Binary Manifest` of the `Software Cluster Connection` shall provide exactly one instance of the `Subscribed Interface Handle Column`

```
const  SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = { <initialization> };
```

⌋

[SWS_SwCluC_00033]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013

⌈The Binary Manifest of the Software Cluster Connection shall allocate and initialize the number of handles in the Subscribed Interface Handle Column according to [SWS_SwCluC_00088].⌋

[SWS_SwCluC_00034]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00009, SRS_SwCluC_00002

⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Subscribed Interface Connected SwClusterId Column

```
const  SwCluC_BManif_SwClusterIdType
    SwCluC_BManif_SubscribedInterfaceConnectedSwClusterIdColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = { <initialization> };
```

⌋

[SWS_SwCluC_00035]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00009, SRS_SwCluC_00002

⌈The Binary Manifest of the Software Cluster Connection shall allocate and initialize the number of handles in the Subscribed Interface Connected SwClusterId Column according to [SWS_SwCluC_00088].⌋

7.1.4.7 Administrative Data

7.1.4.7.1 Immutable Tables Checksum

The Immutable Tables Checksum is built over all those constants of the Binary Manifest that are not changed by the Software Cluster connection step.

This includes

- the Binary Manifest Header
- all arrays of the Interface Descriptor Table
- all arrays of the Offered Interface
- the Subscribed Interface Default Handle Column

The checksum is created as part of the software build.

[SWS_SwCluC_00036]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Immutable Tables Checksum

```
const uint32 SwCluC_BManif_ImmutableTablesChecksum = <initialization>;
```

⌋

[SWS_SwCluC_00037]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The Immutable Tables Checksum shall be set to the value of the configuration parameter SwCluCBManifImmutableTablesChecksum.⌋

[SWS_SwCluC_CONSTR_00073]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The Immutable Tables Checksum shall be calculated on the binary representation in memory of the immutable memory area (inclusive reserved memory space) in ascending order of memory address as CRC32 according to 32-bit Ethernet CRC Calculation as described in document [14].⌋

7.1.4.7.2 Subscribed Interface Validity Marker

The Subscribed Interface Validity Marker indicates that all subscribed tables are written after the Software Cluster connection step. It needs to be set to the valid value after the Software Cluster connection step. The invalid /valid values are not standardized, since those values need to be chosen according to the flash technology storing the Binary Manifest.

[SWS_SwCluC_00040]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Subscribed Interface Validity Marker

```
const uint32 SwCluC_BManif_SubscribedInterfaceValidityMarker = <
    initialization>;
```

⌋

[SWS_SwCluC_00041]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈Initially the `Subscribed Interface Validity Marker` shall be set to the value of the configuration parameter `SwCluCBManifSubscribedInterfaceValidity-Marker`.⌋

[SWS_SwCluC_00095]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈After a completed connection process, the `Subscribed Interface Validity Marker` shall be set to the value `0xFFA5A500`.⌋

7.1.4.8 Memory Mapping

[SWS_SwCluC_00079]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014

⌈The `Binary Manifest` of the `Software Cluster Connection` shall use the `<feature>` = `BMANIF` according to [SWS_MemMap_00040] of document [15].⌋

[SWS_SwCluC_00080]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014

⌈The `Binary Manifest` of the `Software Cluster Connection` shall map the `Binary Manifest Header` to a constant, 32 bit aligned memory section named `CONST_IMMUTABLE_HEADER[_{safety}]_32`.

The optional name part [_{safety}] shall be set according [SWS_SwCluC_00097].⌋

Example 7.1

```
#define SWCLUC_BMANIF_START_SEC_CONST_IMMUTABLE_HEADER_ASIL_D_32
#include "SwCluC_MemMap.h"

const SwCluC_BManif_HeaderType SwCluC_BManif_Header = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_IMMUTABLE_HEADER_ASIL_D_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00081]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014

⌈The `Binary Manifest` of the `Software Cluster Connection` shall map the immutable columns of all tables to a constant, 32 bit aligned memory section named `CONST_IMMUTABLE_COLUMNS[_{safety}]_32`.

The optional name part [_{safety}] shall be set according [SWS_SwCluC_00097].

The immutable columns are:

- all columns of the Interface Descriptor Table
- Offered Interface Handle Column
- Subscribed Interface Default Handle Column

⌋

Example 7.2

```
#define SWCLUC_BMANIF_START_SEC_CONST_IMMUTABLE_COLUMNS_ASIL_D_32
#include "SwCluC_MemMap.h"

const SwCluC_BManif_ResourcePropertiesType
    SwCluC_BManif_ResourcePropertiesDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = ...;

...

const SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceDefaultHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_IMMUTABLE_COLUMNS_ASIL_D_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00082]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014

⌈The Binary Manifest of the Software Cluster Connection shall map the modifiable columns to a constant, 32 bit aligned memory section named CONST_MODIFIABLE_COLUMNS[_{safety}]_32.

The optional name part [_{safety}] shall be set according [SWS_SwCluC_00097].

The modifiable columns are

- Subscribed Interface Handle Column
- Subscribed Interface Connected SwClusterId Column

.⌋

Example 7.3

```
#define SWCLUC_BMANIF_START_SEC_CONST_MODIFIABLE_COLUMNS_ASIL_D_32
#include "SwCluC_MemMap.h"

const SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = ...;

const SwCluC_BManif_SwClusterIdType
    SwCluC_BManif_SubscribedInterfaceConnectedSwClusterIdColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = ...;
```

```
7
8 #define SWCLUC_BMANIF_STOP_SEC_CONST_MODIFIABLE_COLUMNS_ASIL_D_32
9 #include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00083]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014

⌈The Binary Manifest of the Software Cluster Connection shall map the Immutable Tables Checksum to a constant, 32 bit aligned memory section named CONST_IMMUTABLE_TABLES_CHECKSUM[_{safety}]_32.

The optional name part [_{safety}] shall be set according [SWS_SwCluC_00097].⌋

Rationale: Checksums handling in memory might need a fixed location.

Example 7.4

```
1 #define
      SWCLUC_BMANIF_START_SEC_CONST_IMMUTABLE_TABLES_CHECKSUM_ASIL_D_32
2 #include "SwCluC_MemMap.h"
3
4 const uint32 SwCluC_BManif_ImmutableTablesChecksum = ...;
5
6 #define
      SWCLUC_BMANIF_STOP_SEC_CONST_IMMUTABLE_TABLES_CHECKSUM_ASIL_D_32
7 #include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00085]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014

⌈The Binary Manifest of the Software Cluster Connection shall map the Subscribed Interface Validity Marker to a constant, 32 bit aligned memory section named CONST_SUBSCRIBED_INTERFACE_VALIDITY_MARKER[_{safety}]_32.

The optional name part [_{safety}] shall be set according [SWS_SwCluC_00097].⌋

Rationale: Validity handling in memory might need a fixed location.

Example 7.5

```
1 #define
      SWCLUC_BMANIF_START_SEC_CONST_SUBSCRIBED_TABLES_MARKER_ASIL_D_32
2 #include "SwCluC_MemMap.h"
3
4 const uint32 SwCluC_BManif_SubscribedInterfaceValidityMarker = ...;
5
6 #define SWCLUC_BMANIF_STOP_SEC_CONST_SUBSCRIBED_TABLES_MARKER_ASIL_D_32
7 #include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00097]

Upstream requirements: SRS_SwCluC_00014

⌈The name part {safety} shall contain the safety integrity level with at most one of the strings `QM`, `ASIL_A`, `ASIL_B`, `ASIL_C`, `ASIL_D`. In case of `QM` the complete name part [_{safety}] may be omitted.⌋

Please note: The name part `{safety}` is vendor specific and set according the highest supported safety integrity level of the product.

See also [SWS_MemMap_00037] in document [15].

7.1.5 Connecting Software Clusters

7.1.5.1 Overview

The AUTOSAR Methodology supports two different approaches to connect the `Binary Objects` of the different `Software Clusters`.

- The `Software Clusters` are connected after the individual and unconnected `Binary Objects` are programmed on the target ECU, here called `On-board`.
- The `Software Clusters` are connected before the `Binary Objects` are programmed on the target ECU outside the target `machine`, here called `Off-board`.

In the `On-board` case, the `On-board Software Cluster Connector` is a software, which is executed on the target ECU during the reprogramming phase. After reprogramming one or more unconnected `Binary Objects`, the `Software Cluster Connector` stores the connection result into the modifiable area of the `Subscribed Interface` of the `Binary Manifest`.

Please note: The design of the `Binary Manifest` assumes that a connection phase might get aborted at any point in time (e.g. power loss), and a restart or recovery of the connection phase should be supported.

In the `Off-board` case, the `Off-board Software Cluster Connector` is a tool, which is used to read the files of the `Binary Objects`, and to store the result of the connection in files. When reprogramming the ECU, already connected `Binary Objects` are used. Such an `Off-board` process supports additionally the variant to store meta data contained in the `Interface Descriptor Table` outside the `Binary Objects`, with the benefit that no ROM for this meta data is occupied.

Most requirements given in the section are kept universal, and can be applied for the `On-board` and `Off-board` case. If requirements are specific to one case, this is indicated in the requirement. Since AUTOSAR does not standardize the architecture of a flash boot loader, no further interfaces for the `On-board Software Cluster Connector` are standardized.

Some general remarks on `machines`:

Direct connections between Software Clusters based on the Binary Manifest are only made locally (on the same machine). Connecting Software Clusters across 'virtual' machines on the same micro controller is not possible, as it would break the separtion between the virtual machines.

7.1.5.2 Connecting Resources

[SWS_SwCluC_00043]

Upstream requirements: SRS_SwCluC_00002, SRS_SwCluC_00205

⌈The Software Cluster Connector shall connect a required resource with a provided resource, if all of the following conditions are fulfilled:

- the machine Id of the Software Cluster providing the resource and the machine Id of the Software Cluster(s) requesting the resource are identical
- the global resource Id of provided resource and required resource(s) are equal
- the global resource Id is not 0
- the resource type Id of provided resource and required resource is equal
- the guard value of provided resource and required resource is equal
- the number of Provide Handles of provided resource and required resource is equal
- the number of Notifier Handles of provided resource and required resource is equal
- in case Notifier Handles exist, the provided resource has a Notifier Handle Sets available for the required resource

⌋

Note: The global resource Id = 0 is reserved to indicate the absence of a valid Id. The Software Cluster Connector shall only connect if the global resource Id != 0.

7.1.5.3 Handling of the Substitution Software Cluster

The purpose of the Substitution Software Cluster is to temporarily overwrite the output of a Software Cluster for development purposes. For instance, if a sender receiver signal 'Y' of Software Cluster A does not have the right functional behavior, it is possible to add a Substitution Software Cluster which provides a fixed version of sender receiver signal 'Y'. The fixed version of sender receiver signal 'Y' is provided with the identical global resource Id. In such a case, all Software Clusters requiring the sender receiver signal 'Y' are getting connected to the Substitution Software Cluster instead of the original providing Software Cluster. If the Substitution Software Cluster additionally requires the sender receiver

signal 'Y', it gets connected to the original providing Software Cluster. With this principle, it is possible to take the original version of the signal as a basis for the overwriting one.

The utilization of a Substitution Software Cluster is only intended for development purposes, and not supported in productive systems. It is intended as a quick means for problem solving during development (for example, if a signal that is provided by a third party needs to be modified). At a machine, at most one Substitution Software Cluster is supported to avoid ambiguity about the effective Resource Provider.

[SWS_SwCluC_CONSTR_00087]

Upstream requirements: SRS_SwCluC_00005

⌈At a machine, at most one Substitution Software Cluster shall exist.⌋

[SWS_SwCluC_00054]

Upstream requirements: SRS_SwCluC_00005

⌈In case a resource is provided by the Substitution Software Cluster, the Software Cluster Connector shall connect all Software Clusters requiring the resource, except the Substitution Software Cluster itself, to the resource of the Substitution Software Cluster. This happens, regardless if another Software Cluster provides the resource as well. Nevertheless, the matching criteria defined in [SWS_SwCluC_00043] apply.⌋

[SWS_SwCluC_00055]

Upstream requirements: SRS_SwCluC_00005

⌈In case a resource is required by a Substitution Software Cluster, the Software Cluster Connector shall connect this required resource only to provided resources of Software Clusters not being a Substitution Software Cluster. Nevertheless, the matching criteria defined in [SWS_SwCluC_00043] apply.⌋

7.1.5.4 Multiple Notifier Sets

Note: A provided resource can optionally have the ability to handle multiple notifier sets. In this case, the number of handle sets in the Interface Descriptor Table indicates how many handle sets the Software Cluster Connector can register at most. The first handle in the Subscribed Interface Handle Column provides the number of actual registered Notifier Handle Sets. The subsequent handles in the Subscribed Interface Handle Column are store N sets of handles in a consecutive order. The order of the different handle sets written by the Software Cluster Connector are determined by the Software Cluster Id.

[SWS_SwCluC_00045]

Upstream requirements: SRS_SwCluC_00004

⌈In case the provided resource supports multiple Notifier Handle Sets, the Software Cluster Connector shall sort the Notifier Handle Sets in ascending order of Software Cluster Id of the according required resource(s), and place them in the Subscribed Interface Handle Column of the providing Software Cluster, starting at the second handle.⌋

[SWS_SwCluC_00046]

Upstream requirements: SRS_SwCluC_00004

⌈In case the provided resource supports multiple Notifier Handle Sets, the Software Cluster Connector shall write as first entry the actual number of used Notifier Handle Sets in the Subscribed Interface Handle Column.⌋

Note: the actual number of used Notifier Handle Sets is equal to the actual number of connected required resources, since each resource occupies one Notifier Handle Set.

[SWS_SwCluC_00089]

Upstream requirements: SRS_SwCluC_00004

⌈In case the provided resource supports multiple Notifier Handle Sets, the Software Cluster Connector shall write as first entry in the Subscribed Interface Connected SwClusterId Column the value 0xFE.⌋

Note: the value 0xFE indicates that the corresponding cell in the Subscribed Interface Handle Column holds the actual number of used notifier sets.

[SWS_SwCluC_00047]

Upstream requirements: SRS_SwCluC_00004

⌈In case the provide resource supports multiple Notifier Handle Sets, and less required resources are connected, the Software Cluster Connector shall fill the remaining handles in the Subscribed Interface Handle Column with the default values from the Subscribed Interface Default Handle Column of the Software Cluster providing the resource.⌋

The example in table 7.2 shows a connection of a Resource Provider offering at most 4 Notifier Handle sets. The resource type uses two Notifier Handles. The provided resource is connected to two different Software Clusters, and the remaining two Notifier Handle sets are filled with defaults.

Subscribed Interface Handle Column	Subscribed Interface Connected SwClusterId Column	remark
2	0xFE	actual number of used notifier sets, entry in Subscribed Interface Connected SwClusterId Column is set to 0
0xABCDABCD	2	first handle of required resource in Software Cluster 2
0xABC04711	2	second handle of required resource in Software Cluster 2
0xDADA0010	5	first handle of required resource in Software Cluster 5
0xDADA0020	5	second handle of required resource in Software Cluster 5
0xC0CAC01A	0xFF	first default handle of Resource Provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED
0xADD511FE	0xFF	second default handle of Resource Provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED
0xC0CAC01A	0xFF	first default handle of Resource Provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED
0xADD511FE	0xFF	second default handle of Resource Provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED

Table 7.2: Example of multiple notifier set connection

7.1.5.5 Unconnected Resources

[SWS_SwCluC_00044]

Upstream requirements: SRS_SwCluC_00009

⌈The Software Cluster Connector shall copy the Subscribed Interface Default Handle Column entries to the Subscribed Interface Handle Column, and set the related entries in the Subscribed Interface Connected SwClusterId Column to SWCLUC_BMANIF_UNCONNECTED for

- all Provide Handles of any required resources, for which no connect was applied

  AND

- all Notifier Handles of any provided resources, for which no connect was applied

⌋

7.1.5.6 Disabling of the On-board Software Cluster Connection

In general, it is possible that an ECU is equipped with an On-board Software Cluster Connector but this connector should not be used or shall only be used on specific machines.

One reason to do so is a configuration where the software on one machine requires a certification applied on the whole machine's software whereas the software on the other machine can be updated cluster-wise.

For example, such a configuration uses two machines on the same microcontroller, here called Machine A and Machine B.

Machine A needs a certification, which is only granted for the whole software on the machine. Nevertheless, the software is developed by multiple parties, with the means of Software Clusters. The Software Clusters still needs to be connected, which is done with the Off-board Software Cluster Connector before the certification is done.

For Machine B, this restriction does not apply. Here, the On-board Software Cluster Connector shall be used to do some in-field partial updates which requires a working On-board Software Cluster Connector. Since the flash-boot-loader for the whole microcontroller still has to work, the On-board Software Cluster Connector is disabled for Machine A, in order to protect it from unintended changes, caused by an On-board Software Cluster Connector run intended for Machine B.

[SWS_SwCluC_00056]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014

⌈If the bit SWCLUC_BMANIF_DISABLE_ON_ECU_CONNECTION in the ConnectorControlFlags of the Binary Manifest Header of the Host Software Cluster is set, the Software Cluster Connection On-board for this machine is not executed.⌋

7.1.5.7 Errors during software cluster connection

In some cases, the information inside the Binary Manifests might be incompatible with the cluster connector, be incomplete or inconsistent. For example, reading a Binary Manifest, which is structurally incompatible to the Software Cluster Connector's implementation, may have completely undefined results. This section lists a set of conditions that lead to an abort of the connection process.

The behavior of the system after an abort is currently not specified. The implementation of recovery mechanisms is advisable.

[SWS_SwCluC_00048]

Upstream requirements: SRS_SwCluC_00212

⌈The Software Cluster Connector shall detect, if the ManifestMajorVersion and ManifestMinorVersion numbers given in the Binary Manifest Header are compatible to the Software Cluster Connector's implementation. In case of any incompatible version number, the connection process for this specific machine shall be aborted.⌋

If the Software Cluster Ids are not unique on the same machine, the Subscribed Interface Connected SwClusterId Column cannot be set unambiguous:

[SWS_SwCluC_00049]

Upstream requirements: SRS_SwCluC_00212

⌈The Software Cluster Connector shall detect, if the Software Cluster Ids of all Software Clusters belonging to same machine are not unique. In this case, the connection process for this specific machine shall be aborted⌋

If a Software Cluster requires a mandatory resource that no other Software Cluster provides, the operation of the Software Cluster is not possible.

[SWS_SwCluC_00050]

Upstream requirements: SRS_SwCluC_00212

⌈The Software Cluster Connector shall detect, if for a Software Cluster any mandatory required resource is not provided by any other Software Cluster. In this case, the connection process for this specific Software Cluster shall be aborted.⌋

In general, a resource shall only be provided once. The only exception is a second provision by the Substitution Software Cluster, which overrides the provision by another Software Cluster.

[SWS_SwCluC_00051]

Upstream requirements: SRS_SwCluC_00212

⌈The Software Cluster Connector shall detect, if for a machine a resource with the identical resource type id and identical global resource Id is provided twice by Software Clusters, none of which is a Substitution Software Cluster. In this case, the connection process for this specific machine shall be aborted.⌋

[SWS_SwCluC_00052]

Upstream requirements: SRS_SwCluC_00212

⌈In case the resource does not support multiple notifier sets, but uses Notifier Handles, the Software Cluster Connector shall detect if for a machine a resource with the identical resource type id and identical global resource Id is required more than one time. In this case, the connection process for this specific machine shall be aborted.⌋

[SWS_SwCluC_00053]

Upstream requirements: SRS_SwCluC_00212

⌈In case the resource does support multiple notifier sets, the Software Cluster Connector shall detect if for a machine a resource with the identical resource type id and identical global resource Id is required more often as notifier sets are available. In this case, the connection process for this specific machine shall be aborted.⌋

7.1.6 Software Cluster Binary Manifest Descriptor

The Software Cluster Binary Manifest Descriptor describes the Binary Manifest in the Binary Object. This is mandatory for Off-board connection, but also supporting the On-board connection, since it enables to apply checks outside the target ECU.

[SWS_SwCluC_00086]

Upstream requirements: SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00212

⌈The Binary Manifest of the Software Cluster Connection shall provide one instance of the CpSoftwareClusterBinaryManifestDescriptor, which holds the description of the generated Binary Manifest. This includes

- the cpSoftwareCluster reference to the given CpSoftwareCluster
- the softwareClusterId of the given CpSoftwareCluster
- the metaDataFields for IMMUTABLE_TABLES_CHECKSUM and SUBSCRIBED_INTERFACE_VALIDITY_MARKER
- a provideResources for each Provide Resource Entry
- a requireResources for each Require Resource Entry
- a resourceDefinitions for each Resource Type

⌋

7.1.7 Error Classification

On source code level, the implementation of the Binary Manifest is a set of constant tables accessible with a set of C-Macros. Those in turn need to be capable to be static C-initializers. Since the constant tables cannot change at runtime, any implementation of run-time detecting error code is not useful. Therefore, the following sections do not define error codes, and are marked as not applicable.

7.1.7.1 Development Errors

Development errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.1.7.2 Runtime Errors

Runtime errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.1.7.3 Production Errors

Production Errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.2 Software Cluster Base Configuration Check

The Software Cluster Base Configuration Check provides a basic mechanism, to avoid that Application Software Clusters with severe configuration incompatibilities are connected to the Host Software Cluster. This mechanism utilizes the guard value and mandatory feature of the Binary Manifest. The configuration information collected in the two parameters SwCluCAutoBaseConfigDescriptor and SwCluCUserBaseConfigDescriptor is used to calculate the guard value for a mandatory connection.

In case of deviating configurations in the Host Software Cluster and Application Software Cluster, the connection fails. In this case, the whole Application Software Cluster stays unconnected, and is skipped in the connection process (treated, as if it was not present). Please note as well [SWS_SwCluC_00050].

It is implementation dependent, which configuration settings are automatically collected into the SwCluCAutoBaseConfigDescriptor. But it is strongly recommended, to consider only those ECU / machine wide settings that are functional critical, in order to avoid frequent incompatibilities. For instance, compiler settings impacting the EAIB compatibility, like fill byte usage, stack usage, machine register usage.

[SWS_SwCluC_00075]

Upstream requirements: SRS_SwCluC_00212

⌈In the Host Software Cluster, the Software Cluster Base Configuration Check of the Software Cluster Connection shall create one Provide Resource Entry for each configured SwCluCBaseConfigurationCheck, and add it to the Binary Manifest.⌋

[SWS_SwCluC_00076]

Upstream requirements: SRS_SwCluC_00212

⌈In the Application Software Cluster, the Software Cluster Base Configuration Check of the Software Cluster Connection shall create one **mandatory** Require Resource Entry for each configured SwCluCBaseConfigurationCheck, and add them to the Binary Manifest.⌋

[SWS_SwCluC_00077]

Upstream requirements: SRS_SwCluC_00205, SRS_SwCluC_00212

⌈The SwCluCBManifResourceGuardValue of the Resource Entrys of [SWS_SwCluC_00075] and [SWS_SwCluC_00076] shall be calculated out of the given

multi line strings in SwCluCAutoBaseConfigDescriptor and SwCluCUserBaseConfigDescriptor.⌋

7.3 Cross Cluster Communication

7.3.1 Overview

The Cross Cluster Communication implements the VFB communication between Software Clusters. For this purpose, the Cross Cluster Communication implements a Cross Software Cluster Communication Plug-In, as specified in document [4].

Due to the intended separation of Software Clusters (e.g. well separated build units, independent timing), some VFB communication features would either become very inefficient or would break the intended separation. Hence, this specification selected a subset of the most common VFB features expected for such systems, as explained in document [1].

Those are in brief:

- 1:n last is the best sender-receiver communication
- n:1 queued sender-receiver communication
- n:1 client server communication - asynchronous server calls only
- 1:n mode switch communication
- 1:n parameter communication
- limited applicability of PortInterfaceMappings, see 7.3.2.2
- DataReceivedEvents are polled
- AsynchronousServerCallReturnsEvent are polled
- OperationInvokedEvents for Cross Cluster Communication calls are polled

Please note, polling RTEEvents mean, that those events are mapped to periodically activated OS Tasks and the RTE implements the activation of the RunnableEntitys in combination with the Cross Software Cluster Communication Plug-In. (See [SWS_Rte_82017], [SWS_Rte_82018])

When designing an implementation of a Cross Cluster Communication, this intended separation has to be considered. Since each Software Clusters is built by its own, only a limited knowledge about the other Software Clusters is in place. It is also not the target of this Cross Cluster Communication to break the goal of independent development of Software Clusters by exchange of detailed Software Clusters internal design and configuration information.

Instead, the code design patterns have to deal with limited knowledge about the communication counter part.

Due to the limited knowledge about the communication partner, the execution of code might fail in partitioned systems.

[SWS_SwCluC_CONSTR_03066]

Upstream requirements: SRS_SwCluC_00100

⌈The implementation of a Cross Cluster Communication of an Application Software Clusters shall not execute code belonging to any other Software Cluster, except from special entry points provided by the Host Software Cluster (e.g. to implement event passing)⌋

For this reason, SynchronousServerCallPoints are not supported for Cross Cluster Communication. Please note as well [SWS_Rte_72023] in document [4].

In addition, inside a partitioned system it is not guaranteed that the RAM is writable for each Software Cluster or any given partition of a Software Cluster.

[SWS_SwCluC_CONSTR_03067]

Upstream requirements: SRS_SwCluC_00100

⌈The implementation of a Cross Cluster Communication of an Application Software Cluster shall only write into the memory of other Software Clusters, if that write does not cross a Partition border. To determine this, the Application Software Cluster needs to determine if the CpSoftwareClusterResource is assigned to the same partition.⌋

Please note: It might be required to restrict some communication patterns to intra-partition communication, for example to reach a certain performance. But this decreases the overall flexibility in the design and integration of Software Clusters, and therfore should only be done in rare, exceptional cases.

7.3.2 General Requirements

7.3.2.1 Cross Software Cluster Communication Plug-In

[SWS_SwCluC_03000]

Upstream requirements: SRS_SwCluC_00100

⌈The Cross Cluster Communication of the Software Cluster Connection shall implement a Cross Software Cluster Communication Plug-In according to document [4].⌋

[SWS_SwCluC_03000] means among other things:

[SWS_SwCluC_03001]

Upstream requirements: SRS_SwCluC_00100

⌈The `Vendor Specific Module Definition` (see document [16]) of the `Software Cluster Connection` shall provide a definition of the container `RteRipsPluginProps` as a `subContainer` of the container `SwCluCXcc` with the `shortName` `RteRipsPluginProps`.⌋

Please note that the `shortName` of the `EcucContainerValue` related to the `RteRipsPluginProps` container can be freely chosen. This name will be used to determine the name part `<PlugIn>` in the `RTE Implementation Plug-In Services` according to [SWS_Rte_70034].

[SWS_SwCluC_CONSTR_03069]

Upstream requirements: SRS_SwCluC_00100, SRS_BSW_00167

⌈The `EcucModuleConfigurationValues` of the `SwCluC` module shall set the `RteRipsPluginCommunicationScope` to `RTE_RIPS_CROSS_SW_CLUSTER_COM`.⌋

Note: It is possible to use multiple `Cross Software Cluster Communication Plug-In`s in one `Software Cluster`, but this might be rarely used in practice.

Additionally, the `Cross Cluster Communication` of the `Software Cluster Connection` has to provide all the `RTE Implementation Plug-In Services` according to the given input configuration.

[SWS_SwCluC_03002]

Upstream requirements: SRS_SwCluC_00100

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall implement the `RTE Implementation Plug-In` functionality for a `Communication Graph`, if

- a `FlatInstanceDescriptor` references the `Communication Graph`

  AND

- and the `rtePluginProps.associatedCrossSwClusterComRtePlugin` references the `RteRipsPluginProps` container of **this** `Cross Cluster Communication` implementation.

⌋

7.3.2.2 Cross Cluster Communication and Binary Manifest

[SWS_SwCluC_03003]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00212

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall put one `Resource Entry` for each associated `Communication Graph` into the `Binary Manifest`.⌋

[SWS_SwCluC_03004]

Upstream requirements: SRS_SwCluC_00100

⌈The `SwCluCBManifResourceRef` of the `Resource Entry` shall reference the `CpSoftwareClusterCommunicationResource` which is mapped by the `PortElementToCommunicationResourceMapping` to the `Communication Graph`.⌋

[SWS_SwCluC_03005]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00205

⌈The `SwCluCBManifResourceGuardValue` of the `Resource Entry` shall be calculated out of the following properties:

- data range
- resolution
- physical meaning
- encoding
- structure in memory

⌋

[SWS_SwCluC_03033]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00205

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall normalize the numerical values of attributes, before using them for the `SwCluCBManifResourceGuardValue` calculation, in such a way that compatible numerical values according to [TPS_GST_02501] result in the identical guard value.⌋

Note: [SWS_SwCluC_03033] shall ensure that the calculated guard value result is independent from the number format (e.g. decimal, binary, octal and hexadecimal) of the given attributes. For instance the values, 1, 1.0, 0x000001 are semantically equivalent, and have to result in the identical guard value contribution.

Additionally, the below requirements on the guard value calculation in some cases define default values for optional existing attributes. This ensures an identical guard value on provider an requester side, even if one side expresses the default value explicitly and the other side omits it.

Only in special cases, the shortName of AUTOSAR model elements has an explicit semantic. Hence, shortNames are not relevant for the guard value, except if it is explicitly specified.

The list of attributes relevant for the guard value, specified in the following requirements, guarantee that different implementations of Software Cluster Connections treat the same set of required and provided ports as compatible, if these conditions apply:

- the elements in the PortInterfaces are compatible according to document [17] without PortInterfaceMappings
- or if a PortInterfaceMappings would be applied, it does not result in
  - Data Scaling

    NOR
  - utilization of a SubElementMapping

    NOR
  - utilization of a ClientServerOperationMapping

    NOR
  - utilization of a ClientServerApplicationErrorMapping

    NOR
  - utilization of a ModeDeclarationMappingSet

Please note:

In case the following requirements about guard value calculation list references as attributes, the calculation has to consider the attributes of the referenced model element. The fully qualified shortName (building the reference) is not part of the guard value. For example, the attribute requiredInterface is a reference to a PortInterface. In this case, the mentioned attributes of the referenced PortInterface are relevant instead of the fully qualified shortName (e.g. /Example/PortInterfaces/MySRInterface)

7.3.2.2.1 Guard value calculation for Data Communication Graphs

[SWS_SwCluC_03034]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the SwCluCBManifResourceGuardValue calculation for a Data Communication Graph:

- PPortPrototype.providedInterface
- RPortPrototype.requiredInterface
- CpSoftwareClusterCommunicationResource.communicationResourceProps.sendIndication
  - in case of SenderReceiverInterface
    - VariableDataPrototype.type
    - VariableDataPrototype.swDataDefProps
    - SenderReceiverInterface.invalidationPolicy.handleInvalid (if existing for the VariableDataPrototype, else dontInvalidate applies)
  - in case of ParameterInterface
    - ParameterDataPrototype.type
    - ParameterDataPrototype.swDataDefProps

  Additionally, attributes of VariableDataPrototype / ParameterDataPrototype.type according to [SWS_SwCluC_03035] and [SWS_SwCluC_03036] apply.

⌋

7.3.2.2.2 Guard value calculation for Client Server Communication Graphs

[SWS_SwCluC_03047]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00103, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the SwCluCBManifResourceGuardValue calculation for a Client Server Communication Graph:

- PPortPrototype.providedInterface
- RPortPrototype.requiredInterface
  - ClientServerOperation.arguments {ordered}
    - ArgumentDataPrototype.type
    - ArgumentDataPrototype.direction
  - ClientServerOperation.possibleError.errorCode (if existing, else one errorCode = 0 is applied)

    Additionally, attributes of ArgumentDataPrototype.type according to [SWS_SwCluC_03035] and [SWS_SwCluC_03036] apply.

⌋

7.3.2.2.3 Guard value calculation for Mode Communication Graphs

[SWS_SwCluC_03045]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00104, SRS_SwCluC_00205

⌈The `Cross Cluster Communication` shall use the following attributes for the `SwCluCBManifResourceGuardValue` calculation for a `Mode Communication Graph`:

- `PPortPrototype.providedInterface`
- `RPortPrototype.requiredInterface`
- `ModeDeclarationGroupPrototype.type`
- `ModeDeclarationGroup.category`
- `ModeDeclarationGroup.initialMode`
- `ModeDeclarationGroup.modeDeclaration`s
- `ModeDeclarationGroup.onTransitionValue` (in case `ModeDeclarationGroup.category` == `EXPLICIT_ORDER`)
- `ModeDeclarationGroup.modeUserErrorBehavior` (if existing)
- `ModeDeclarationGroup.modeManagerErrorBehavior` (if existing)
- `ModeDeclarationGroup.modeTransition`s (if existing) for each `ModeDeclaration` the following attributes apply:
  - `ModeDeclaration.shortName`
  - `ModeDeclaration.value` (in case `ModeDeclarationGroup.category` == `EXPLICIT_ORDER`)

  for each `ModeErrorBehavior` the following attributes apply:
  - `ModeErrorBehavior.errorReactionPolicy`
  - `ModeErrorBehavior.defaultMode` (if existing)

  for each `ModeTransition` the following attributes apply:
  - `ModeTransition.exitedMode`
  - `ModeTransition.enteredMode`

⌋

7.3.2.2.4 Guard value calculation for Trigger Communication Graphs

[SWS_SwCluC_03046]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00105, SRS_SwCluC_00205

⌈The `Cross Cluster Communication` shall use the following attributes for the `SwCluCBManifResourceGuardValue` calculation for a `Trigger Communication Graph`:

- `PPortPrototype.providedInterface`
- `RPortPrototype.requiredInterface`
- `Trigger.swImplPolicy` (if existing, else `standard` applies)

⌋

7.3.2.2.5 Guard value calculation for DataPrototypes

[SWS_SwCluC_03035]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The `Cross Cluster Communication` shall use the following attributes for the referenced `ApplicationDataType` for the `SwCluCBManifResourceGuardValue` calculation:

- in case the `AutosarDataPrototype.type` refers to an `ApplicationPrimitiveDataType`
  - `ApplicationPrimitiveDataType.category`
  - `ApplicationPrimitiveDataType.swDataDefProps`
- in case the `AutosarDataPrototype.type` refers to an `ApplicationArrayDataType`
  - `ApplicationArrayDataType.dynamicArraySizeProfile` (if existing)
  - `ApplicationArrayDataType.element`
  - `ApplicationArrayElement.type`
  - `ApplicationArrayElement.arraySizeSemantics` (if existing, else `fixedSize` applies)
  - `ApplicationArrayElement.arraySizeHandling` (if existing)
  - `ApplicationArrayElement.maxNumberOfElements`
- in case the `AutosarDataPrototype.type` refers to an `ApplicationRecordDataType`

  - ApplicationRecordDataType.elements {ordered}
  - ApplicationRecordElement.type
  - ApplicationRecordElement.isOptional (if existing, else false applies)

Additionally, the attributes of the mapped ImplementationDataType according to [SWS_SwCluC_03036] and the attributes of SwDataDefProps according to [SWS_SwCluC_03037] apply.⌋

[SWS_SwCluC_03036]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the referenced or mapped ImplementationDataType for the SwCluCBManifResourceGuardValue calculation:

- in case the AutosarDataPrototype.type refers to an ImplementationDataType of category TYPE_REFERENCE or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category TYPE_REFERENCE
  - ImplementationDataType.swDataDefProps

  in case the AutosarDataPrototype.type refers to an ImplementationDataType of category VALUE or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category VALUE
  - ImplementationDataType.swDataDefProps
- in case the AutosarDataPrototype.type refers to an ImplementationDataType of category ARRAY or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category ARRAY
  - ImplementationDataType.dynamicArraySizeProfile (if existing)
  - ImplementationDataType.swDataDefProps
  - ImplementationDataType.subElement
  - ImplementationDataTypeElement.arraySizeSemantics (if existing, else fixedSize applies)
  - ImplementationDataTypeElement.arraySizeHandling (if existing)
  - ImplementationDataTypeElement.swDataDefProps
- in case AutosarDataPrototype.type refers to an ImplementationDataType of category STRUCTURE or the ApplicationDataType according to

[SWS_SwCluC_03036] is mapped to an ImplementationDataType of category STRUCTURE

- ImplementationDataType.swDataDefProps
- ImplementationDataType.isStructWithOptionalElement (if existing, else false applies)
- ImplementationDataType.subElement {ordered}
- ImplementationDataTypeElement.arraySizeSemantics (if existing, else fixedSize applies)
- ImplementationDataTypeElement.arraySizeHandling (if existing)
- ImplementationDataTypeElement.swDataDefProps

Additionally, the attributes of SwDataDefProps according to [SWS_SwCluC_03037] apply.

⌋

[SWS_SwCluC_03037]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the given SwDataDefProps for the SwCluCBManifResourceGuardValue calculation:

- SwDataDefProps.additionalNativeTypeQualifier (if existing)
- SwDataDefProps.swImplPolicy (if existing, else standard applies)
- SwDataDefProps.invalidValue (if existing)
- SwDataDefProps.unit (if existing, else CompuMethod.unit applies)
- SwDataDefProps.dataConstr (if existing)
- SwDataDefProps.compuMethod (if existing)
- SwDataDefProps.swTextProps (if existing)
- SwDataDefProps.baseType (if existing)
- SwDataDefProps.swValueBlockSize (if existing)
- SwDataDefProps.swValueBlockSizeMult (if existing)
- SwDataDefProps.valueAxisDataType (if existing)
- SwDataDefProps.implementationDataType (if existing)

Here, the prioritization of SwDataDefProps attributes according to [constr_1015] applies. Only the most prior SwDataDefProps attributes are considered.

Additionally, the attributes of Unit, CompuMethod, DataConstr, and BaseType according to [SWS_SwCluC_03038], [SWS_SwCluC_03039], [SWS_SwCluC_03042], [SWS_SwCluC_03043], and [SWS_SwCluC_03044] apply.

⌋

[SWS_SwCluC_03038]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the given Unit for the SwCluCBManifResourceGuardValue calculation:

- Unit.factorSiToUnit (if existing, else 1 applies)
- Unit.offsetSiToUnit (if existing, else 0 applies)
- Unit.physicalDimension (if existing, else [SWS_SwCluC_03041] applies)

Additionally, the attributes of PhysicalDimension according to [SWS_SwCluC_03040] and [SWS_SwCluC_03041] apply.⌋

[SWS_SwCluC_03039]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈In case **no** Unit is given for the SwCluCBManifResourceGuardValue calculation, the Cross Cluster Communication shall use the following attribute values:

- Unit.factorSiToUnit = 1
- Unit.offsetSiToUnit = 0
- Unit.physicalDimension according to [SWS_SwCluC_03041]

Additionally, the attributes of PhysicalDimension according to [SWS_SwCluC_03041] apply.⌋

[SWS_SwCluC_03040]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the given PhysicalDimension for the SwCluCBManifResourceGuardValue calculation:

- PhysicalDimension.shortName
- PhysicalDimension.currentExp (if existing, else 0 applies)
- PhysicalDimension.lengthExp (if existing, else 0 applies)
- PhysicalDimension.luminousIntensityExp (if existing, else 0 applies)
- PhysicalDimension.massExp (if existing, else 0 applies)

- PhysicalDimension.molarAmountExp (if existing, else 0 applies)
- PhysicalDimension.temperatureExp (if existing, else 0 applies)
- PhysicalDimension.timeExp (if existing, else 0 applies)

⌋

[SWS_SwCluC_03041]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈In case **no** PhysicalDimension is given, the Cross Cluster Communication shall use the following attribute values for the SwCluCBManifResourceGuardValue calculation:

- PhysicalDimension.shortName = NoDimension
- PhysicalDimension.currentExp = 0
- PhysicalDimension.lengthExp = 0
- PhysicalDimension.luminousIntensityExp = 0
- PhysicalDimension.massExp = 0
- PhysicalDimension.molarAmountExp = 0
- PhysicalDimension.temperatureExp = 0
- PhysicalDimension.timeExp = 0

⌋

The guard value calculation has to consider the intended compatibility of CompuMethod with different category, as defined in document [17] with [constr_1176] and [constr_1192]. Please note as well [constr_1375] w.r.t the existence of attributes of CompuMethods. Nevertheless, not all attributes are considered for the guard value, in order to increase compatibility between provider and requesters.

[SWS_SwCluC_03042]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the given CompuMethod for the SwCluCBManifResourceGuardValue calculation:

- CompuMethod.unit
- in case the CompuMethod.category is IDENTICAL
  - it is treated like a CompuMethod where the category is LINEAR where the CompuMethod.compuPhysToInternal.compuScale yield the conversion

$int = \frac{N_0 + N_1 * phys}{D_0}$ if $N_0 \sim 0$ && $N_1 \sim 1$ && $D_0 \sim 0$

- in case the CompuMethod.category is LINEAR
  - CompuMethod.compuPhysToInternal.compuScale

    OR
  - CompuMethod.compuInternalToPhys.compuScale

    where the compuInternalToPhys coefficients are converted to equivalent compuPhysToInternal coefficients.
    - CompuScale.compuNumerator.vs {ordered}
    - CompuScale.compuDenominator.v
- in case the CompuMethod.category is SCALE_LINEAR
  - CompuMethod.compuPhysToInternal.compuScales
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    OR
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    - CompuScale.upperLimit
    - CompuScale.lowerLimit
    - CompuScale.compuNumerator.vs {ordered}
    - CompuScale.compuDenominator.v
- in case the CompuMethod.category is RAT_FUNC with 2 compuNumerator coefficients and 1 compuDenominator coefficient
  - this case is treated like a CompuMethod where the category is LINEAR
- in case the CompuMethod.category is RAT_FUNC not matching the LINEAR case
  - CompuMethod.compuPhysToInternal.compuScale
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    AND
  - CompuMethod.compuInternalToPhys.compuScale

  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuNumerator.vs {ordered}
    * CompuScale.compuDenominator.v {ordered}
- in case the CompuMethod.category is SCALE_RAT_FUNC
  - CompuMethod.compuPhysToInternal.compuScales
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    AND
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuNumerator.vs {ordered}
    * CompuScale.compuDenominator.v {ordered}
- in case the CompuMethod.category is TEXTTABLE
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt
- in case the CompuMethod.category is BITFIELD_TEXTTABLE
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)

    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.mask
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt

- in case the CompuMethod.category is SCALE_LINEAR_AND_TEXTTABLE
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.mask
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt (if existing)
    * CompuScale.compuNumerator.vs {ordered} (if existing)
    * CompuScale.compuDenominator.v (if existing)
- in case the CompuMethod.category is SCALE_RATIONAL_AND_TEXTTABLE
  - CompuMethod.compuPhysToInternal.compuScales
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    AND
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.mask
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt (if existing)
    * CompuScale.compuNumerator.vs {ordered} (if existing)
    * CompuScale.compuDenominator.v {ordered} (if existing)

- in case the CompuMethod.category is TAB_NOINTP
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    - CompuScale.upperLimit
    - CompuScale.lowerLimit
    - CompuScale.compuInverseValue.vf (if existing)
    - CompuScale.compuConst.vt (if existing)

  Additionally, the attributes of Unit according to [SWS_SwCluC_03038], [SWS_SwCluC_03039] apply.

⌋

[SWS_SwCluC_03043]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The Cross Cluster Communication shall use the following attributes for the given DataConstr for the SwCluCBManifResourceGuardValue calculation:

- DataConstr.dataConstrRule.physConstrs.upperLimit (if existing)
- DataConstr.dataConstrRule.physConstrs.lowerLimit (if existing)
- DataConstr.dataConstrRule.physConstrs.monotony (if existing)
- DataConstr.dataConstrRule.physConstrs.maxGradient (if existing)
- DataConstr.dataConstrRule.physConstrs.maxDiff (if existing)
- DataConstr.dataConstrRule.internalConstrs.upperLimit (if existing, else based on baseTypeEncoding and baseTypeSize the technical upper limit is taken)
- DataConstr.dataConstrRule.internalConstrs.lowerLimit (if existing, else based on baseTypeEncoding and baseTypeSize the technical lower limit is taken)
- DataConstr.dataConstrRule.internalConstrs.monotony (if existing, else noMonotony)
- DataConstr.dataConstrRule.internalConstrs.maxGradient (if existing)
- DataConstr.dataConstrRule.internalConstrs.maxDiff (if existing)

⌋

[SWS_SwCluC_03044]

Upstream requirements: SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106, SRS_SwCluC_00205

⌈The `Cross Cluster Communication` shall use the following attributes for the given `SwBaseType` for the `SwCluCBManifResourceGuardValue` calculation:

- `SwBaseType.baseTypeEncoding`
- `SwBaseType.baseTypeSize`
- `SwBaseType.byteOrder`
- `SwBaseType.nativeDeclaration`
- `SwBaseType.memAlignment` (if existing, else `0` applies)

⌋

7.3.2.3 Cross Cluster Communication Base Socket

In general, it is possible that the communication between `Application Software Clusters` needs some basic infrastructure from the `Host Software Cluster`. For instance, to pass an activity or event between `Application Software Clusters`. Usually, such event passing requires some knowledge about the overall environment, which is **NOT** available in an `Application Software Cluster`. For instance, in the `Application Software Cluster`, only a subset of the defined partitions (and related `EcucPartition`s plus `OsApplication`s) are known.

[SWS_SwCluC_03012]

Upstream requirements: SRS_SwCluC_00212

⌈If the implementation of a `Cross Cluster Communication` requires some basic infrastructure from the `Host Software Cluster`, it shall use the means of the `SwCluCXccBaseSocket` configuration to link to the resource pool.⌋

[SWS_SwCluC_03013]

Upstream requirements: SRS_SwCluC_00212

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall put one `Resource Entry` for each associated `SwCluCXccBaseSocket` into the `Binary Manifest`.⌋

7.3.2.4 Cross Cluster Communication and McSupport Data

[SWS_SwCluC_03071]

Upstream requirements: SRS_SwCluC_00300

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall provide an *MC-Support* (Measurement and Calibration) description as part of its *Basic Software Module Description*.⌋

[SWS_SwCluC_03072]

Upstream requirements: SRS_SwCluC_00300

⌈The `McSupportData` element begin part of [SWS_SwCluC_03071] and its sub-structure shall be self-contained, in the sense that there is no need to deliver the whole upstream descriptions of the ECU (including the `ECU Extract`, software component descriptions, Basic Software Module descriptions, ECU Configuration Values descriptions, etc.), in order to later generate the final "A2L"-file. This means that the `Cross Cluster Communication` generator has to copy the required information from the upstream descriptions into the `McSupportData` element.⌋

[SWS_SwCluC_03073]

Upstream requirements: SRS_SwCluC_00300

⌈The `Cross Cluster Communication` generator shall export the effective `SwDataDefProps` (including all of the referenced and aggregated sub-elements like e.g. `CompuMethod` or `SwRecordLayout`) in the role `resultingProperties`, for each `McDataInstance`, after resolving the precedence rules defined in the SW-Component Template [17] Chapter 5.4 *"Properties of Data Definitions"*. Thereby, the `ImplementationDataType` properties `compuMethod` and `dataConstr` are not taken in consideration for effective `SwDataDefProps` of the `McDataInstance`, due to their refinement nature of **C** and **AI**.⌋

[SWS_SwCluC_03074]

Upstream requirements: SRS_SwCluC_00300

⌈The `Cross Cluster Communication` generator shall export one entry in the `McSupportData`, describing the default data instance according to [SWS_SwCluC_03120], with the role `mcParameterInstance`, with the following attributes:

- `swCalibrationAccess` set to `readWrite`
- effective `SwDataDefProps` according to [SWS_SwCluC_03073]
- `McDataInstance.subElement`s for array elements or structure elements, if applicable
- `symbol` set to the C-symbol name used for the default data instance

if

- the parameter SwCluCXccSwCluCXccDefaultDataHandling is set to DEFAULTS_AS_CONSTANTS,

  AND

- the effective swCalibrationAccess according to [SWS_SwCluC_03073] of the related VariableDataPrototype is set to set to readOnly or readWrite

⌋

[SWS_SwCluC_03077]

Upstream requirements: SRS_SwCluC_00300

⌈The Cross Cluster Communication generator shall export one entry in the McSupportData, describing the default data instance according to [SWS_SwCluC_03120], with the role mcParameterInstance, with the following attributes:

- swCalibrationAccess set to readOnly
- effective SwDataDefProps according to [SWS_SwCluC_03073]
- McDataInstance.subElements for array elements or structure elements, if applicable
- symbol set to the C-symbol name used for the default data instance

if

- the parameter SwCluCXccSwCluCXccDefaultDataHandling is set to DEFAULTS_AS_CONSTANTS

  AND

- the effectiveSwDataDefProps according to [SWS_SwCluC_03073] of the related VariableDataPrototype is set to readOnly or readWrite

⌋

[SWS_SwCluC_03075]

Upstream requirements: SRS_SwCluC_00300

⌈The Cross Cluster Communication generator shall export one entry in the McSupportData, describing the default parameter instance according to [SWS_SwCluC_03006], with the role mcParameterInstance, with the following attributes:

- swCalibrationAccess set to readWrite
- effective SwDataDefProps according to [SWS_SwCluC_03073]
- McDataInstance.subElements for array elements or structure elements, if applicable
- symbol set to the C-symbol name used for the default parameter instance

⌋

[SWS_SwCluC_03076]

Upstream requirements: SRS_SwCluC_00300

⌈If the `swCalibrationAccess` of a `VariableDataPrototype` instance in the delegation `PPortPrototype` at the `CompositionSwComponentType` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract` is set to `readOnly` or `readWrite`, the `Cross Cluster Communication` generator shall export one entry in the `McSupportData`, describing the data instance according to [SWS_SwCluC_03101], with the role `mcVariableInstance`, with the following attributes:

- effective `SwDataDefProps` (inclusive `swCalibrationAccess`) according to [SWS_SwCluC_03073]
- `McDataInstance`.`subElement`s for array elements or structure elements, if applicable
- `symbol` set to the C-symbol name used for the data instance
- if applicable, `flatMapEntry` referencing to the corresponding `FlatInstanceDescriptor` element of the `VariableDataPrototype`

⌋

7.3.3 Sender Receiver Communication

7.3.3.1 Restrictions on VFB communication features

[SWS_SwCluC_03064]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall reject the configuration, if there is a “N:1” sender-receiver communication with last-is-the-best semantic.⌋

[SWS_SwCluC_03065]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall reject the configuration, if there is a “1:N” sender-receiver communication with event semantic (queued communication).⌋

[SWS_SwCluC_03068]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The implementation of a `Cross Cluster Communication` shall support that `Data Communication Graphs` between `Application Software Clusters` can be added, without modification of the `Host Software Cluster`.⌋

7.3.3.2 Data Concistency

[SWS_SwCluC_03084]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00107

⌈If the attribute DataComProps.dataConsistencyPolicy is set to consistencyMechanismRequired for the Data Communication Graph, the Cross Cluster Communication of the Software Cluster Connection shall use an appropriate mechanism to guarantee data consistency if data inconsistency could occur.⌋

Please note that such a consistency mechanism is implemented preferable with a lock free pattern, since in a clustered ECU software the optimization of semaphores is barely possible.

[SWS_SwCluC_03085]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00107

⌈If the attribute DataComProps.dataConsistencyPolicy is set to noConsistencyMechanism for the Data Communication Graph, the Cross Cluster Communication of the Software Cluster Connection shall not use any mechanisms to guarantee data consistency.⌋

Rationale: If it is guaranteed on system level, that the data exchange on that Data Communication Graph will not cause any data inconsistencies, noConsistencyMechanism can be set. Thus, no extra data consistency mechanism is required to be used by the Software Cluster Connection.

7.3.3.3 Transmission

[SWS_SwCluC_03101]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The Cross Cluster Communication of the Software Cluster Connection shall provide one data instance, for each assigned Data Communication Graph

- where the referenced VariableDataPrototype is owned by a delegation PPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract AND
- where implicit communication applies to that VariableDataPrototype OR
- where explicit unqueued communication applies to that VariableDataPrototype

⌋

[SWS_SwCluC_03102]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The data instance created according to [SWS_SwCluC_03101] shall be mapped to a VAR memory section, according to document [15].⌋

[SWS_SwCluC_03103]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall make the `data instance` created according to [SWS_SwCluC_03101], accessible to other `Software Cluster`s via a `Resource Entry` in the `Binary Manifest`.⌋

[SWS_SwCluC_03106]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When implicit or explicit unqueued communication applies, the `Rte_Rips_Write` shall update the data instance created according to [SWS_SwCluC_03101]), and return `RTE_E_OK`.⌋

[SWS_SwCluC_03142]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00212

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall provide one `Resource Entry`, for each assigned `Data Communication Graph`

- where the referenced `VariableDataPrototype` is owned by a delegation `PPortPrototype` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract` AND
- where explicit queued communication applies to that `VariableDataPrototype`

This `Resource Entry` shall be used to link the access of the related `Rte_Rips_Write` service to the according queue instance on the receiving `Software Cluster`.⌋

[SWS_SwCluC_03107]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is not connected, the `Rte_Rips_Write` shall discard the write access, and return `RTE_E_OK`.⌋

[SWS_SwCluC_03108]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is connected, the `Rte_Rips_Write` shall check

the queue status (See [SWS_SwCluC_03135]), to determine whether or not the data can be enqueued.⌋

[SWS_SwCluC_03109]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is connected, and the queue is not full, the `Rte_Rips_Write` shall enqueue the data and return `RTE_E_OK` (See [SWS_SwCluC_03135]).⌋

[SWS_SwCluC_03110]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is connected, but the queue is full, the `Rte_Rips_Write` shall discard the write access and return `RTE_E_LIMIT` (See [SWS_SwCluC_03135]).⌋

[SWS_SwCluC_03080]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When implicit or explicit unqueued communication applies and the attribute `sendIndication` is set to `anySendOperation` for the `Data Communication Graph`, the call of the `Rte_Rips_Write` shall indicate the occurrence of a send operation for the related data instance.⌋

This send operation indication can be used at the receiving `Software Cluster`s to determine update flags or data receive event triggers.

Please note that the specification behavior supports a polling of those send indications and therefore an active passing of those events can be avoided.

The value of the `sendIndication` attribute is only relevant for unqueued communication (last-is-the best), since the implementation of a queue on the receiver side provides in any case the possibility to detect enqueue operations.

[SWS_SwCluC_03111]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `data instance` created according to [SWS_SwCluC_03101] shall be initialized according to the `NonqueuedSenderComSpec.initValue` in the `PPortPrototype` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract`.⌋

[SWS_SwCluC_03112]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈If acknowledgment is enabled on a `Data Communication Graph` where explicit unqueued communication applies, and the resource created according to

[SWS_SwCluC_03103] is not connected, the Rte_Rips_Feedback API shall return RTE_E_UNCONNECTED.⌋

[SWS_SwCluC_03113]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈If acknowledgement is enabled on a Data Communication Graph where explicit unqueued communication applies, and the resource created according to [SWS_SwCluC_03103] is connected, the Rte_Rips_Feedback API shall return RTE_E_TRANSMIT_ACK.⌋

[SWS_SwCluC_03405]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈If acknowledgment is enabled on a Data Communication Graph where explicit queued communication applies, and the resource created according to [SWS_SwCluC_03142] is not connected, the Rte_Rips_Feedback API shall return RTE_E_UNCONNECTED.⌋

[SWS_SwCluC_03406]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈If acknowledgement is enabled on a Data Communication Graph where explicit queued communication applies, and the resource created according to [SWS_SwCluC_03142] is connected and the data could be enqueued, the Rte_Rips_Feedback API shall return RTE_E_TRANSMIT_ACK.⌋

7.3.3.4 Reception

[SWS_SwCluC_03120]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The Cross Cluster Communication of the Software Cluster Connection shall provide one default data instance, for each assigned Data Communication Graph

- where the referenced VariableDataPrototype is owned by a delegation RPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract AND
- where implicit communication applies to that VariableDataPrototype OR
- where explicit unqueued communication applies to that VariableDataPrototype

⌋

[SWS_SwCluC_03133]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall provide a queue data instance (where the queue size corresponds to the specified queue length), for each assigned `Data Communication Graph`

- where the referenced `VariableDataPrototype` is owned by a delegation `RPortPrototype` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract` AND
- where explicit queued communication applies to that `VariableDataPrototype`

⌋

[SWS_SwCluC_03134]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall determine the queue size according to the `queueLength` attribute of the `QueuedReceiverComSpec`⌋

[SWS_SwCluC_03135]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When [SWS_SwCluC_03133] applies, the `Cross Cluster Communication` of the `Software Cluster Connection` shall provide a data instance, representing the number of possible entries in the queue.⌋

[SWS_SwCluC_03122]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈If the parameter `SwCluCXccSwCluCXccDefaultDataHandling` is set to `DEFAULTS_AS_CONSTANTS`, the default data instance created according to [SWS_SwCluC_03120] shall be mapped to a `CONST` memory section, according to document [15].⌋

[SWS_SwCluC_03121]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈If the parameter `SwCluCXccSwCluCXccDefaultDataHandling` is set to `DEFAULTS_AS_CALPRMS`, the default data instance created according to [SWS_SwCluC_03120] shall be mapped to a `CALPRM` memory section, according to document [15].⌋

Please note as well that section 7.3.2.4 is relevant for calibration parameters instantiated by `Software Cluster Connection`.

[SWS_SwCluC_03123]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The data instances created according to [SWS_SwCluC_03120], [SWS_SwCluC_03133] and [SWS_SwCluC_03135] shall be mapped to a VAR memory section, according to document [15].⌋

[SWS_SwCluC_03124]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The default data instance created according to [SWS_SwCluC_03120] shall be initialized according to the `NonqueuedReceiverComSpec.initValue` in the `RPortPrototype` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract`.⌋

[SWS_SwCluC_03126]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall use the default data instance created according to [SWS_SwCluC_03120] as default handle in the corresponding `Resource Entry` in the `Binary Manifest`.⌋

[SWS_SwCluC_03136]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall make the data instance created according to [SWS_SwCluC_03133] accessible from other `Software Clusters`, via a `Resource Entry` in the `Binary Manifest`.⌋

[SWS_SwCluC_03130]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈When explicit unqueued communication via `dataReceivePointByArgument`s applies, and the `Resource Entry` created according to[SWS_SwCluC_03126] is not connected, the `Rte_Rips_Read` shall copy the value of default data instance created according to [SWS_SwCluC_03120] to the location of the OUT parameter <data>, and shall return `RTE_E_UNCONNECTED`.⌋

[SWS_SwCluC_03131]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈When explicit unqueued communication via `dataReceivePointByValue`s applies, and the `Resource Entry` created according to [SWS_SwCluC_03126] is not connected, the `Rte_Rips_DRead` shall return the value of default data instance created according to [SWS_SwCluC_03120].⌋

[SWS_SwCluC_03132]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When explicit unqueued communication via dataReceivePointByArguments applies, and the Resource Entry created according to [SWS_SwCluC_03126] is connected, the Rte_Rips_Read shall copy the value of the corresponding data instance of the provider to the location of the OUT parameter <data>, and shall return RTE_E_OK.⌋

[SWS_SwCluC_03137]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈When explicit queued communication applies, and the resource created according to [SWS_SwCluC_03136] is not connected, the Rte_Rips_Read shall discard the read access, and return RTE_E_UNCONNECTED.⌋

[SWS_SwCluC_03138]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When explicit queued communication applies, and the Resource Entry created according to [SWS_SwCluC_03136] is connected, but no entry is available in the queue, the Rte_Rips_Read shall discard the read access, and return RTE_E_NO_DATA .⌋

[SWS_SwCluC_03139]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When explicit queued communication applies, and the Resource Entry created according to [SWS_SwCluC_03136] is connected, the Rte_Rips_Read shall copy the value of the first available entry of the corresponding data instance to the location of the OUT parameter <data>, and return RTE_E_OK.⌋

[SWS_SwCluC_03140]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈When [SWS_SwCluC_03139] applies, the Rte_Rips_Read shall dequeue the data from the corresponding queue.⌋

[SWS_SwCluC_03143]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101, SRS_SwCluC_00213

⌈If explicit unqueued communication applies, and the Resource Entry in the Binary Manifest created according to [SWS_SwCluC_03126] is not connected, the Rte_Rips_DataIsUpdated shall return FALSE.⌋

[SWS_SwCluC_03144]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈If explicit unqueued communication applies, and the Resource Entry in the Binary Manifest created according to [SWS_SwCluC_03126] is connected, and the

sender has updated the data since the previous execution of the corresponding Rte_Rips_Read service, the Rte_Rips_DataIsUpdated shall return TRUE

⌋

[SWS_SwCluC_03081]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The Rte_Rips_DataIsUpdated shall return TRUE if

- explicit unqueued communication applies

  AND

- the Resource Entry in the Binary Manifest created according to [SWS_SwCluC_03126] is connected

  AND

- the occurrence of a send operation was indicated after previous calls of Rte_Read or Rte_DRead related to this reading software component.

⌋

[SWS_SwCluC_03082]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The Rte_Rips_DataIsUpdated_EventActivation shall return TRUE if

- the Resource Entry in the Binary Manifest created according to [SWS_SwCluC_03126] or [SWS_SwCluC_03136] is connected

  AND

  - the occurrence of a send operation was indicated after previous calls of this instance of the Rte_Rips_DataIsUpdated_EventActivation

    OR

  - an enqueue operation occurred after previous calls of this instance of the Rte_Rips_DataIsUpdated_EventActivation

⌋

7.3.4 NvData communication

7.3.4.1 Restrictions on VFB communication features

Please note that the Cross Cluster Communication of the Software Cluster Connection does not support the NvData communication. This means its not possible that a communication graph involves a NvBlockSwComponent and additionally get mapped to the Cross Software Cluster Communication Plug-In.

This use case can be implemented by a software component copying the data of a NvBlockSwComponent to the Software Cluster's interface using a regular sender receiver communication.

[SWS_SwCluC_CONSTR_03400] Data Communication Graph which involves a NvBlockSwComponent

Upstream requirements: SRS_SwCluC_00100, SRS_BSW_00167

⌈The Cross Cluster Communication of the Software Cluster Connection shall not be configured such that, a Data Communication Graph which involves a NvBlockSwComponent is associated to the Cross Cluster Communication according to [SWS_SwCluC_03002].⌋

7.3.5 Client Server Communication

7.3.5.1 General

[SWS_SwCluC_03145]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall ensure that the result of a ClientServerOperation is dispatched to the correct client, if more than one client invokes the ClientServerOperation⌋

[SWS_SwCluC_03146]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication shall support multiple Clients invoking the same ClientServerOperation on a server ('N:1' Communication where N $\geq$ 1).⌋

7.3.5.2 Timeout

[SWS_SwCluC_03147]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall ensure that timeout monitoring is performed for client-server communication.⌋

[SWS_SwCluC_03087]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall implement the timeout monitoring of a client in the associated SwCluC_Xcc_MainFunction.⌋

[SWS_SwCluC_03088]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈When a timeout occurred, the Cross Cluster Communication of the Software Cluster Connection shall discard any subsequent responses to that request.⌋

Please note, in case a timeout occurs the related Rte_Rips_ReturnResult returns RTE_E_TIMEOUT ([SWS_SwCluC_03169]) and buffers for the IN/OUT and OUT parameters are not be modified ([SWS_Rte_89023], [SWS_Rte_01114]).

[SWS_SwCluC_03089]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall reject configurations where a client with a ServerCallPoint.timeout value > 0 is associated to a SwCluC_Xcc_MainFunction where the ServerCallPoint.timeout value is not an integer multiple of the SwCluCXccMainFunction.SwCluCXccMainTimeBase value.⌋

[SWS_SwCluC_03090]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall associate all clients to a specific SwCluC_Xcc_MainFunction which are explicitly mapped via a SwCluCXccClientInstanceRef of the related SwCluCXccMainFunction container.⌋

[SWS_SwCluC_03091]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall associate all clients which are not referenced explicitly by a SwCluCXccClientInstanceRef with the default SwCluC_Xcc_MainFunction for the EcucPartition on which the SwComponentPrototype owning the client is mapped to.⌋

[SWS_SwCluC_03092]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall reject configurations where a client in an assigned Client Server Communication Graph with a ServerCallPoint.timeout value > 0 exists, which is not associated a SwCluC_Xcc_MainFunction.⌋

[SWS_SwCluC_CONSTR_03093]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_BSW_00167

⌈Clients of a SwComponentPrototype shall only be associated to SwCluC_Xcc_MainFunctions belonging to the identical EcucPartition on which the SwComponentPrototype is mapped to.⌋

7.3.5.3 Scheduling of main functions

[SWS_SwCluC_03094]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈In case SwCluCXccScheduleIntegration is set to SERVICE_SOFTWARE_COMPONENT, the Cross Cluster Communication of the Software Cluster Connection shall create one partition specific Service Software Component per configured EcucPartition for the Xcc, with the name SwCluC_Xcc_<EcucPartition shortName>.⌋

[SWS_SwCluC_03095]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈In case SwCluCXccScheduleIntegration is set to SERVICE_SOFTWARE_COMPONENT, the Cross Cluster Communication of the Software Cluster Connection shall create one RunnableEntity per configured SwCluCXccMainFunction container in the partition specific Service Software Component according [SWS_SwCluC_03094].

The EcucPartition is determined by the parameter SwCluCXccMainPartitionRef. The attributes of the RunnableEntity shall be set as following:

- symbol is set to the name of the SwCluC_Xcc_MainFunction according [SWS_SwCluC_91002]
- canBeInvokedConcurrently is set to false

⌋

[SWS_SwCluC_03096]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_BSW_00478

⌈In case SwCluCXccScheduleIntegration is set to SERVICE_SOFTWARE_COMPONENT, the Cross Cluster Communication of the Software Cluster Connection shall create one TimingEvent per configured SwCluCXccMainFunction container in the partition specific Service Software Component according [SWS_SwCluC_03094].

The attributes of the TimingEvent shall be set as following:

- startOnEvent is set to the RunnableEntity according [SWS_SwCluC_03095]
- period value is set according the SwCluCXccMainTimeBase parameter

⌋

Note: shortName of TimingEvent is vendor specific.

Note: According [SWS_BSW_00001] the SwCluC module has to provide a BSW Module description containing a BswModuleDescription. This general requirement is

not repeated in this specification, but it's the basis for further requirements on descriptive elements.

[SWS_SwCluC_03097]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈In case `SwCluCXccScheduleIntegration` is set to `BASIC_SW_MODULE`, the `Cross Cluster Communication` of the `Software Cluster Connection` shall create one `BswSchedulableEntity` per configured `SwCluCXccMainFunction` container in its `BswModuleDescription`.

The attributes of the `BswSchedulableEntity` shall be set as following:

- `shortName` is set to the name of the `SwCluC_Xcc_MainFunction` according [SWS_SwCluC_91002]
- `reentrancyLevel` is set to `false`

⌋

[SWS_SwCluC_03098]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_BSW_00478

⌈In case `SwCluCXccScheduleIntegration` is set to `BASIC_SW_MODULE`, the `Cross Cluster Communication` of the `Software Cluster Connection` shall create one `BswTimingEvent` per configured `SwCluCXccMainFunction` container in its `BswModuleDescription`.

The attributes of the `BswTimingEvent` shall be set as following:

- `startsOnEvent` is set to the `BswSchedulableEntity` according [SWS_SwCluC_03097]
- `period` value is set according the `SwCluCXccMainTimeBase` parameter

⌋

Note: `shortName` of `BswTimingEvent` is vendor specific.

7.3.5.4 Buffering

[SWS_SwCluC_03150]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` on the client side of the `Software Cluster Connection` shall provide a request buffer for each client of the assigned `Client Server Communication Graph`.

Note: The structure of the request buffer is implementation specific, but the `Cross Cluster Communication` has to store at least the IN parameters and IN/OUT parameters, the transaction handle and the status code of the communication.⌋

[SWS_SwCluC_03152]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall provide a request queue with the specified queue length (See [SWS_SwCluC_03154]), for each assigned Client Server Communication Graph, where the referenced ClientServerOperation is owned by a delegation PPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract.

Note: The structure of the queue is implementation specific, but the Cross Cluster Communication has to store at least the IN/OUT and OUT parameters, the transaction handle and the status code of the communication.⌋

[SWS_SwCluC_03153]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall provide N response buffers (where N corresponds to the specified queue length according [SWS_SwCluC_03154]), for each assigned Client Server Communication Graph, where the referenced ClientServerOperation is owned by a delegation PPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract.

Note: The structure of the response buffer is implementation specific, but the Cross Cluster Communication has to store at least the IN/OUT and OUT parameters, the transaction handle and the status code of the communication.⌋

[SWS_SwCluC_03154]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall determine the queue length according to the following priority rules (highest priority first):

1. value of the ECU-C parameter RteServerQueueLength
2. value of the queueLength attribute of the ServerComSpec

⌋

Note: The queue length can be pre-defined at Software Cluster design time with the attribute ClientServerOperationComProps.queueLength. But it is in the responsibility of the integrator to consider this value for the cluster configuration.

[SWS_SwCluC_03155]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall handle the requests in a first-in-first-out queue.⌋

[SWS_SwCluC_03156]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall make the request buffer instance created according to [SWS_SwCluC_03150] accessible for the `Software Cluster` implementing the server, via a `Resource Entry` in the `Binary Manifest`.⌋

[SWS_SwCluC_03158]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall make each request queue instance created according to [SWS_SwCluC_03152] accessible to the `Software Cluster`, via a `Resource Entry` in the `Binary Manifest`.⌋

[SWS_SwCluC_03159]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00101

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall make each response buffer instance created according to [SWS_SwCluC_03153] accessible to the `Software Clusters`, via a `Resource Entry` in the `Binary Manifest`.⌋

7.3.5.5 Response to Request Mapping

The `Cross Cluster Communication` is responsible to map a response to the corresponding request. The problem of request to response mapping is split into:

- Mapping of a response to the correct client in the `Software Cluster`.
- Mapping of a response to the correct request within one client in the `Software Cluster`.

The general approach for request response mapping is to use transaction handles.

[SWS_SwCluC_03160]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall use a transaction handle that contains three parts, of unsigned integer type:

- Software Cluster Identifier
- Client Identifier
- Client Sequence Counter

⌋

[SWS_SwCluC_03161]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall use the transaction handle for the identification of client server transactions, communicated between the clusters.⌋

[SWS_SwCluC_03162]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` on the server side of the `Software Cluster Connection` shall return the transaction handle of the request, without modification, together with the response.⌋

[SWS_SwCluC_03163]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall allow only one request per client and server operation at any time.⌋

7.3.5.6 Client Side

[SWS_SwCluC_03086]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall treat a client as connected

- if the corresponding resource in the `Binary Manifest` of the request queue (see [SWS_SwCluC_03158])

  AND

- if the corresponding resource in the `Binary Manifest` of the response buffer (see [SWS_SwCluC_03159])

are connected.⌋

Note: This means that if only the request queue or the response buffer gets a connection to the server in the other `Software Cluster`, the client is considered as unconnected.

[SWS_SwCluC_03164]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_SwCluC_00213

⌈If the client is not connected according [SWS_SwCluC_03086], the `Rte_Rips_Invoke` shall return RTE_E_UNCONNECTED immediately.⌋

[SWS_SwCluC_03165]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the client is connected according [SWS_SwCluC_03086], the `Rte_Rips_Invoke` shall copy the IN and IN/OUT parameters, inform the server via a status flag, and return RTE_E_OK.⌋

[SWS_SwCluC_03166]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the client is connected according [SWS_SwCluC_03086], the `Rte_Rips_Invoke` shall return RTE_E_LIMIT, until the server's result has been successfully passed to the client, and no timeout occurred. The IN and IN/OUT parameters shall not be modified by `Rte_Rips_Invoke` .⌋

[SWS_SwCluC_03167]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_SwCluC_00213

⌈If the client is not connected according [SWS_SwCluC_03086], the `Rte_Rips_ReturnResult` shall return RTE_E_UNCONNECTED immediately.⌋

[SWS_SwCluC_03168]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the client is connected according [SWS_SwCluC_03086], the `Rte_Rips_ReturnResult` shall return RTE_E_NO_DATA, until the server's result has been successfully passed to the client and no timeout occurred.⌋

[SWS_SwCluC_03169]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the client is connected according [SWS_SwCluC_03086], and the server's result was not available within the specified timeout, the `Rte_Rips_ReturnResult` shall return RTE_E_TIMEOUT.⌋

[SWS_SwCluC_03170]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_BSW_00172

⌈If the client is connected according [SWS_SwCluC_03086], and an `AsynchronousServerCallReturnsEvent` exists, the `Rte_Rips_InvocationHandler` shall poll the corresponding response queue status to determine whether or not the server's result is available.⌋

[SWS_SwCluC_03171]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the client is connected according [SWS_SwCluC_03086], and an `AsynchronousServerCallReturnsEvent` exists, the `Rte_Rips_InvocationHandler` shall call the corresponding `ASCR runnable`.⌋

7.3.5.7 Server Side

[SWS_SwCluC_03173]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_BSW_00172

⌈If the corresponding resource in the `Binary Manifest` representing the request queue (See [SWS_SwCluC_03158]) is connected, the `Rte_Rips_InvocationHandler` shall poll the corresponding request queue status to determine whether or not there is a pending request.⌋

[SWS_SwCluC_03401]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈One call of the `Rte_Rips_InvocationHandler` shall dequeue pending server call requests until

- either the request queue has no further pending requests

  OR

- it has executed so many server calls as given by the configuration 'maximum number of server invocations' according [SWS_SwCluC_03402].

⌋

[SWS_SwCluC_03402]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall determine the 'maximum number of server invocations' value for a `server runnable` from the parameter `RteServerNumberOfRequestProcessing` of the `RteEventToTaskMapping` applied to the `OperationInvokedEvent` of the `server runnable`.⌋

[SWS_SwCluC_03178]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the

- corresponding resource in the `Binary Manifest` representing the request queue (See [SWS_SwCluC_03158])

  AND

- corresponding resource in the `Binary Manifest` representing the response buffer (See [SWS_SwCluC_03159])

are connected, the `Rte_Rips_InvocationHandler` shall invoke the corresponding `server runnable` with IN, IN/OUT and OUT parameters of request resp. response buffer.⌋

[SWS_SwCluC_03176]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the corresponding resource in the `Binary Manifest` representing the request queue (See [SWS_SwCluC_03158]) is connected, the `Rte_Rips_InvocationHandler` shall dequeue the server queue after successful invocation of the server queue.⌋

[SWS_SwCluC_03177]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈If the corresponding resource in the `Binary Manifest` representing the request queue (See [SWS_SwCluC_03158]) is connected (see [SWS_SwCluC_03086]), the `Rte_Rips_InvocationHandler` shall set the status of the response buffer (See [SWS_SwCluC_03159]) accordingly, to inform the client that the request has been processed.⌋

[SWS_SwCluC_03083]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall ignore server responses with invalid transaction handles on the client side.⌋

For instance the server responds with a transaction handle which does not match to any ongoing client request.

7.3.6 Modes Communication

7.3.6.1 General principles

The mode switch communication in AUTOSAR is a mixture between a data (the mode value) and an event semantic (the transition between modes). It also has some potential impact on the execution of `ExecutableEntity`s, via mode switch events and mode disabling dependencies. Additionally, the transitions between modes are following a strict sequence, as described in document [4].

Introducing this communication pattern to a clustered software architecture requires a technical solution that balances between the local view of software components inside a `Software Cluster`, and the ECU wide behavior cross several `Software Clusters`.

For the cross software cluster communication, the following principles apply:

- The mode providing `Software Cluster` owns the leading `mode machine instance`, including the mode queue.
- The `Host Software Cluster` provides the mode switch tasks, which are used to execute the mode switch on each partition, where the mode is capable to execute runnables

- The Host Software Cluster coordinates the transfer of the mode switch notifications between the Software Clusters.
- The Host Software Cluster coordinates the conjunction of the Software Cluster individual completions of the mode switches, and notifies the mode providing Software Cluster
- Nevertheless, the mode switch itself is locally executed in each RTE in a Software Cluster. This has the consequence that the order between on-exit ExecutableEntitys, on-transition ExecutableEntitys, and on-entry ExecutableEntitys is only preserved locally inside a Software Cluster.
- The interfaces towards the RTE are only called in well-defined OS task contexts.

A sender-receiver communication can be implemented solely between Application Software Clusters, without impact to the Host Software Cluster. In contrast, the adding of a new mode switch communication between Application Software Clusters, additionally impacts the Host Software Cluster.

7.3.6.2 Software Cluster providing a mode

The Software Cluster providing the mode owns the leading mode machine instance, including its mode queue. Hence, the information about the next mode, to which the mode machine currently switches, is provided by this Software Cluster. In this specification, this mode value is called 'current on-transition value'.

The figure 7.3 illustrates the principle sequence, when a mode switch is initiated by an Application Software Cluster.

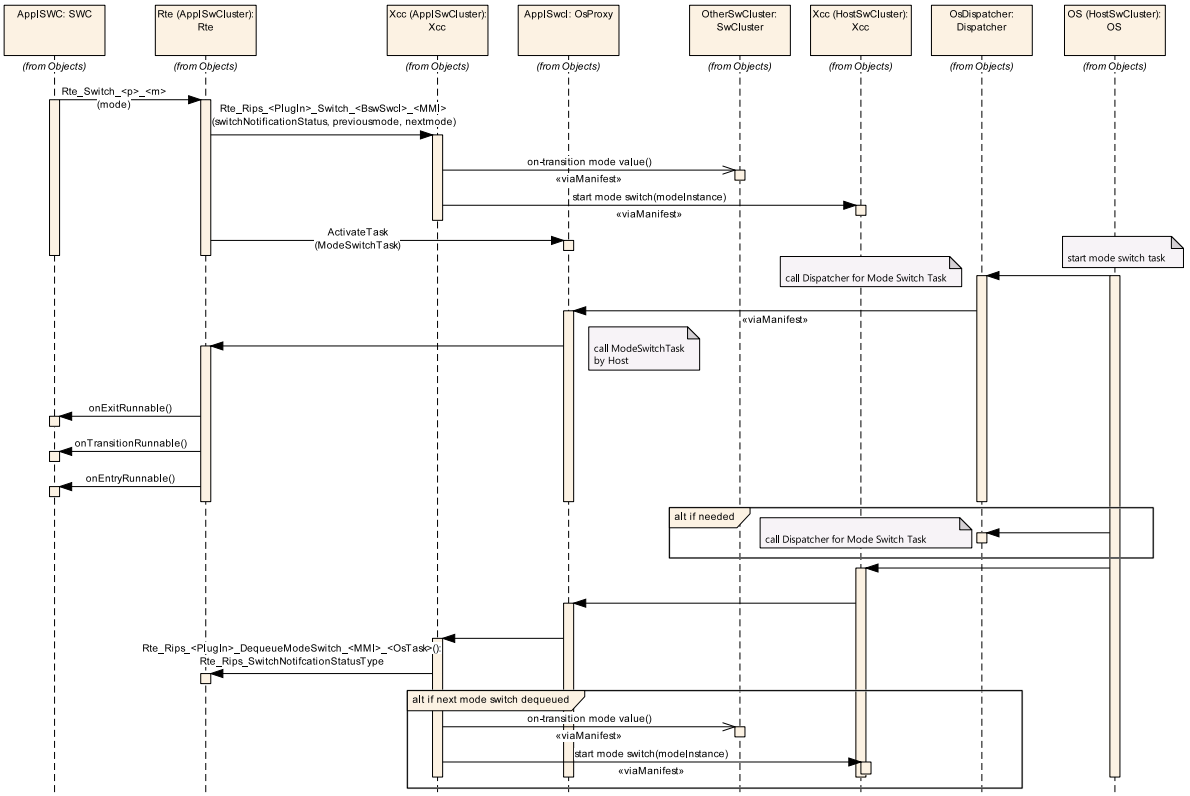

Figure 7.3: Execution of a mode switch provided by an Application Software Cluster

- The mode manager software component calls the Rte_Switch API to do the mode switch notification
- The Cross Cluster Communication gets notified via the call of the Rte_Rips_Switch Service.
- If the mode queue is empty, this Rte_Rips_Switch can be used to determine the first on-transition value.
- The Cross Cluster Communication in the Application Software Cluster informs the Cross Cluster Communication in the Host Software Cluster that a mode switch starts. This notification is a Cross Cluster Communication internal interface, which is not standardized.
- After the start of the mode switch on the host, the according mode switch tasks are scheduled (potentially on multiple partitions). This in turn, via Dispatchers, schedules cluster local proxy mode switch tasks.
- Potentially, other proxy mode switch tasks are scheduled before and after. Hence, the RTE cannot dequeue the next mode already in the context of its mode switch tasks.
- After the Cross Cluster Communication in the Host Software Cluster has determined that a mode switch task has run to end, it uses Cross

Cluster Communication to notify the RTE in the Application Software Cluster to dequeue the next mode.

The figure 7.4 illustrates the principle sequence, when a mode switch is initiated by a Host Software Cluster.

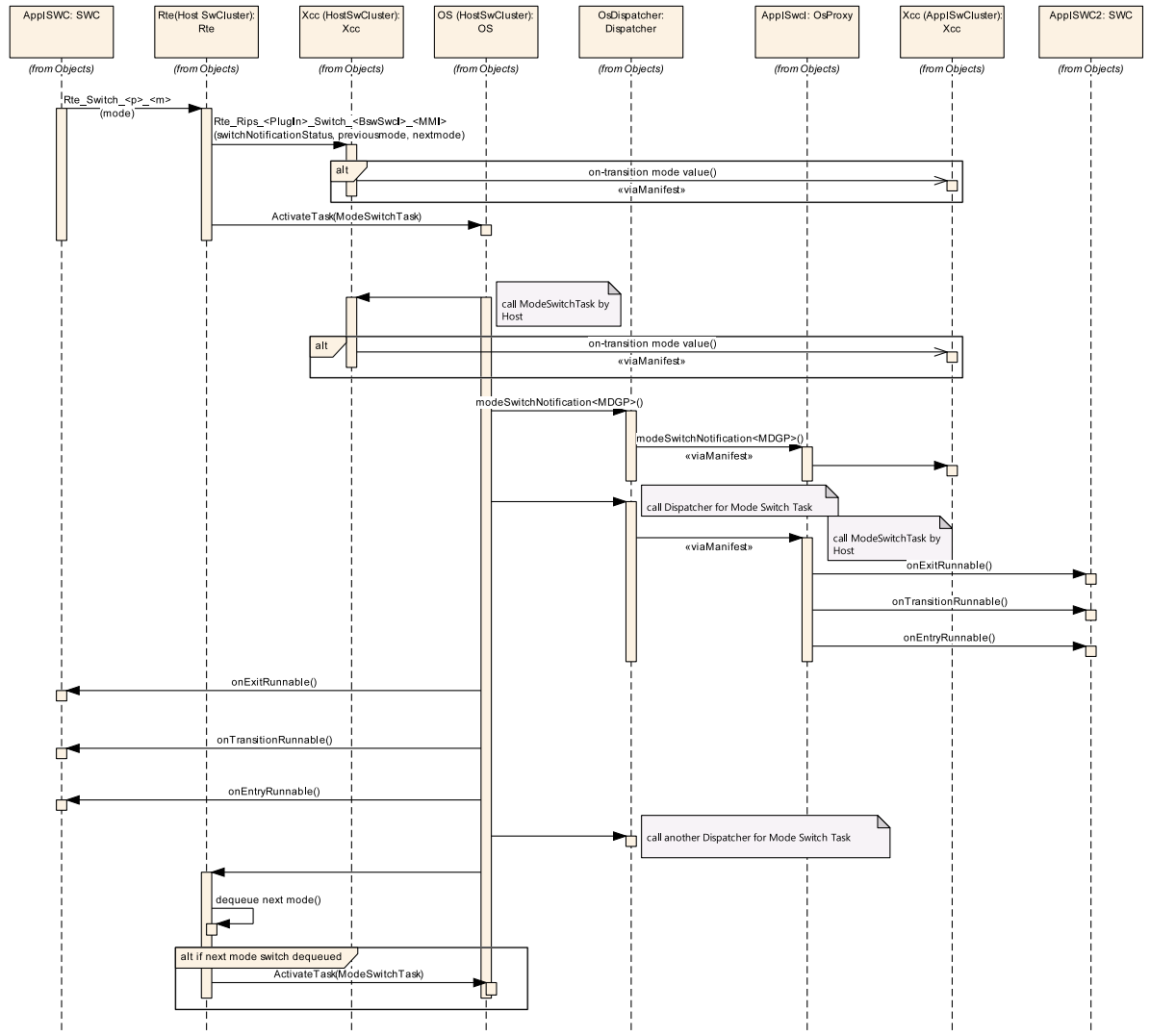

Figure 7.4: Execution of a mode switch provided by an Host Software Cluster

- The mode manager software component calls the Rte_Switch API, to do the mode switch notification
- The Cross Cluster Communication gets notified via the call of the Rte_Rips_Switch Service.
- If the mode queue is empty, this Rte_Rips_Switch can be used to determine the first on-transition value. Alternatively, this is also possible in the mode switch task(s) before first Dispatchers are scheduled.

- After the start of the mode switch on the host, the according mode switch tasks are scheduled (potentially on multiple partitions). This in turn, via Dispatchers, schedules cluster local proxy mode switch tasks.
- In contrast to the scenario in figure 7.3, the RTE can dequeue the next mode already in the context of its mode switch tasks.

These two principle sequences lead to the following requirements on the Cross Cluster Communication, in case a Software Cluster provides a mode.

[SWS_SwCluC_03022]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall signal the current on-transition value to other Software Clusters, before the mode switch according to [SWS_Rte_02665] starts.⌋

Note: This functionality can be implemented inside the Rte_Rips_Switch service, in case the mode queue was empty. Otherwise, this can be done shortly before the last mode switch task terminates, or when the first mode switch task starts already for the next transition. The interface to signal the current on-transition is a Cross Cluster Communication internal interface, which is not standardized.

[SWS_SwCluC_03057]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall add at the Software Cluster providing the mode a Provide Resource Entry, and at any Software Cluster requiring the trigger an according Require Resource Entry in the Binary Manifest.⌋

[SWS_SwCluC_03057] ensures that a missing mode manager in a clustered ECU gets detected, even if the implementation may rely additionally on the according Dispatchers.

[SWS_SwCluC_03023]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication of the Host Software Cluster shall implement a mode manager for each mode which needs to be provided by an Application Software Cluster.⌋

With this mode manager, it is possible to schedule the according Dispatchers for mode switch notification, and Dispatchers for Software Cluster local proxy mode switch tasks.

To ease the realization of [SWS_SwCluC_03023], all modes provided by the Application Software Clusters, shall be consumed by the Host Software Cluster.

[SWS_SwCluC_CONSTR_03032]

Upstream requirements: SRS_SwCluC_00104, SRS_BSW_00167

⌈For each mode provided by an Application Software Cluster, the Ecu Extract of the Host Software Cluster shall own a required mode switch port at the CompositionSwComponentType of the rootSoftwareComposition.⌋

[SWS_SwCluC_03024]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication of the Host Software Cluster shall enqueue the current on-transition values signaled by Application Software Cluster, via the related mode manager. ([SWS_SwCluC_03023])⌋

Note: In case the Host Software Cluster provides the mode, the RTE implements already the leading mode machine instance.

[SWS_SwCluC_03025]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication of the Host Software Cluster shall notify the Application Software Cluster, when a mode switch tasks starts.⌋

Note: This notification is used by the Application Software Cluster to schedule the mode switch notification runnable, which enqueues the mode switch notification in the cluster local RTE.

[SWS_SwCluC_03026]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication of the Host Software Cluster shall notify the Cross Cluster Communication of the Application Software Cluster, when all mode switch tasks run to end.⌋

[SWS_SwCluC_03027]

Upstream requirements: SRS_SwCluC_00104

⌈In case the Application Software Cluster provides the mode, the Cross Cluster Communication of the Host Software Cluster shall notify the Cross Cluster Communication of the Application Software Cluster, when mode switch tasks run to end.⌋

[SWS_SwCluC_03028]

Upstream requirements: SRS_SwCluC_00104

⌈In case the Application Software Cluster provides the mode, the Cross Cluster Communication of the Application Software Cluster notifies the RTE via Rte_Rips_DequeueModeSwitch, when a mode switch task known by this RTE runs to end. The Cross Cluster Communication has to guarantee that the

last call of Rte_Rips_DequeueModeSwitch is not done before the last on-entry ExecutableEntity in the whole clustered system terminated.⌋

Please note [SWS_Rte_70123], which guarantees a certain execution context for the RTE.

[SWS_SwCluC_03029]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide a Complex Driver Software Component on each EcucPartition, where mode switch task are configured. This component is later called mode proxy component.⌋

[SWS_SwCluC_03030]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide RunnableEntitys to detect the start and end of any mode switch task, which is related to mode communication cross Software Clusters.⌋

Please note: the RunnableEntitys [SWS_SwCluC_03030] may require additional ModeAccessPoints, as well as suitable RTEEvents. It is possible to use OsTaskExecutionEvent and a small runtime logic, to determine the ongoing mode transition, or a set of SwcModeSwitchEvents, which activate the RunnableEntity on any transition of a specific mode machine instance.

7.3.6.3 Host Software Cluster requiring a mode

As explained in section 7.3.6.2, the Host Software Cluster is already aware about existing modes communicated cross Software Clusters. Therefore, it technically receives already the modes provided by Application Software Clusters. In case the Host Software Cluster owns software components, which require the the mode as well, it shall connect them to its already existing mode manager.

[SWS_SwCluC_03031]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide the AssemblySwConnectors between the provided mode switch ports of the mode proxy component and the required mode switch ports at the software components requiring the mode, if the mode is required by the Host Software Cluster.⌋

7.3.6.4 Application Software Cluster requiring a mode

For each required mode, the Application Software Cluster defines OsTask(s), for the mode switch notification.

When the mode switch gets notified by the Host Software Cluster, the Cross Cluster Communication writes the actual on-transition mode value to the RTE of the Application Software Cluster with the regular Rte_Switch API related to a mode switch port.

Note: The RTE in the Application Software Cluster shall still a call ActivateTask(s) to trigger the mode switch task(s). This call can still be used by the Host Software Cluster to trigger the mode switch task execution in the Host Software Cluster, if needed.

For this purpose, the Cross Cluster Communication defines a RunnableEntity with an OsTaskExecutionEvent for each mode on each partition where a mode switch notification is configured. When the mode switch is executed, the Host Software Cluster schedules the mode switch task(s), which in turn schedule the 'proxy' mode switch task(s) in the Application Software Cluster

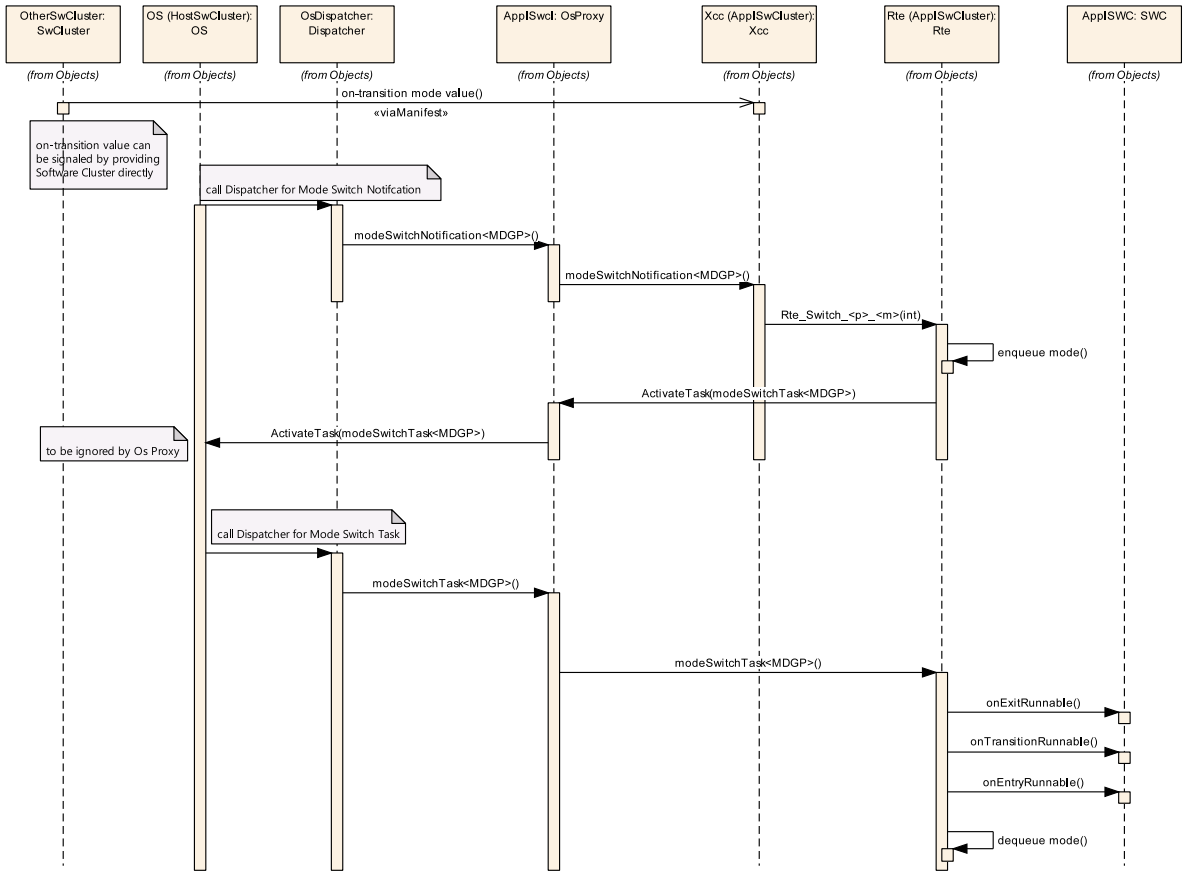

Figure 7.5: Execution of a mode switch in an Application Software Cluster

[SWS_SwCluC_03015]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide a Complex Driver Software Component on each EcucPartition where a mode switch port needs to be provided. This component is later called mode proxy component.⌋

[SWS_SwCluC_03016]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide a provided mode switch port on each mode proxy component where a mode switch port needs to be provided.⌋

[SWS_SwCluC_03017]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide the AssemblySwConnectors between the provided mode switch ports and the required mode switch ports at the software components requiring the mode.⌋

[SWS_SwCluC_03018]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide a mode switch notification runnable with an OsTaskExecutionEvent, for each mode on each partition where a mode switch notification is configured. The RunnableEntity shall define a ModeSwitchPoint.⌋

[SWS_SwCluC_03019]

Upstream requirements: SRS_SwCluC_00104

⌈When the mode switch notification runnable is executed, it shall write the actual on-transition value via Rte_Switch to the RTE.⌋

[SWS_SwCluC_CONSTR_03020]

Upstream requirements: SRS_SwCluC_00104, SRS_BSW_00167

⌈For each EcucPartition on which the mode switch notification is configured for a ModeDeclarationGroupPrototype required by an Application Software Cluster, a OsTask with a related SwCluCOsProxyOsTask shall be configured. On the Host Software Cluster, a SwCluCOsProxyOsTaskDispatcher shall be configured.⌋

[SWS_SwCluC_CONSTR_03021]

Upstream requirements: SRS_SwCluC_00104, SRS_BSW_00167

⌈For each required ModeDeclarationGroupPrototype and EcucPartition on which the mode switch notification is configured, a OsTask with a related

SwCluCOsProxyOsTask and SwCluCOsProxyOsTaskDispatcher shall be configured.⌋

7.3.6.5 Initialization

The approach described in section 7.3.6.2 and 7.3.6.4 leads to local mode machine instances in each Software Cluster providing or requiring the mode.

But this also implies that the initialization of the mode machine instances is executed time shifted, without explicit synchronization by the Cross Cluster Communication.

When the clustered system starts, the following procedure needs to be preserved!

- Initialization of the Cross Cluster Communications
- Execute all Rte_Init_<InitContainer> functions in Application Software Clusters and Host Software Cluster. This step already leads to the execution of on-entry ExecutableEntitys, triggered by initialMode and mapped to RteInitializationRunnableBatch container.
- Execute all Rte_Start in Application Software Clusters. This step executes the remaining on-entry ExecutableEntitys triggered by initialMode. In case the RTE implementation triggers the mode switch task to proceed the transition to the initialMode, it will happen here as well.
- Execute Rte_Start in the Host Software Cluster. This step executes the remaining on-entry ExecutableEntitys triggered by initialMode in the Host Software Cluster. In case the RTE implementation triggers the mode switch task to proceed the transition to the initialMode, the mode switch task will now be executed.

7.3.7 Trigger Communication

7.3.7.1 General principles

The trigger communication in AUTOSAR is a pure event semantic, which is used to request the execution of triggered ExecutableEntitys.

Introducing this communication pattern to a clustered software architecture, the focus lies on the use case to implement trigger sources in Application Software Clusters and in the Host Software Cluster. Nevertheless, the control on the execution of the triggered ExecutableEntitys is implemented in the Host Software Cluster, to guarantee an ECU wide behavior cross several Software Clusters.

For the cross software cluster communication, the following principles apply:

- The trigger providing Application Software Cluster transfers the occurrence of the trigger to the Host Software Cluster.
- The Host Software Cluster provides the tasks, which are used to execute the triggered ExecutableEntitys on each partition where needed.
- If queuing of triggers is required from a dynamic perspective, this is configured and implemented in the RTE of the Host Software Cluster
- Nevertheless, the triggering itself is locally executed in each RTE inside a Software Cluster. This has the consequence that the order between triggered ExecutableEntitys is only preserved locally inside a single proxy OS Task of an Application Software Cluster.
- The interfaces towards the RTE are only called in well-defined OS task contexts.

Adding of a new trigger communication between Application Software Clusters, additionally impacts the Host Software Cluster.

7.3.7.2 Software Cluster providing a trigger

The figure 7.6 illustrates the principle sequence, when a trigger is raised by an Application Software Cluster.

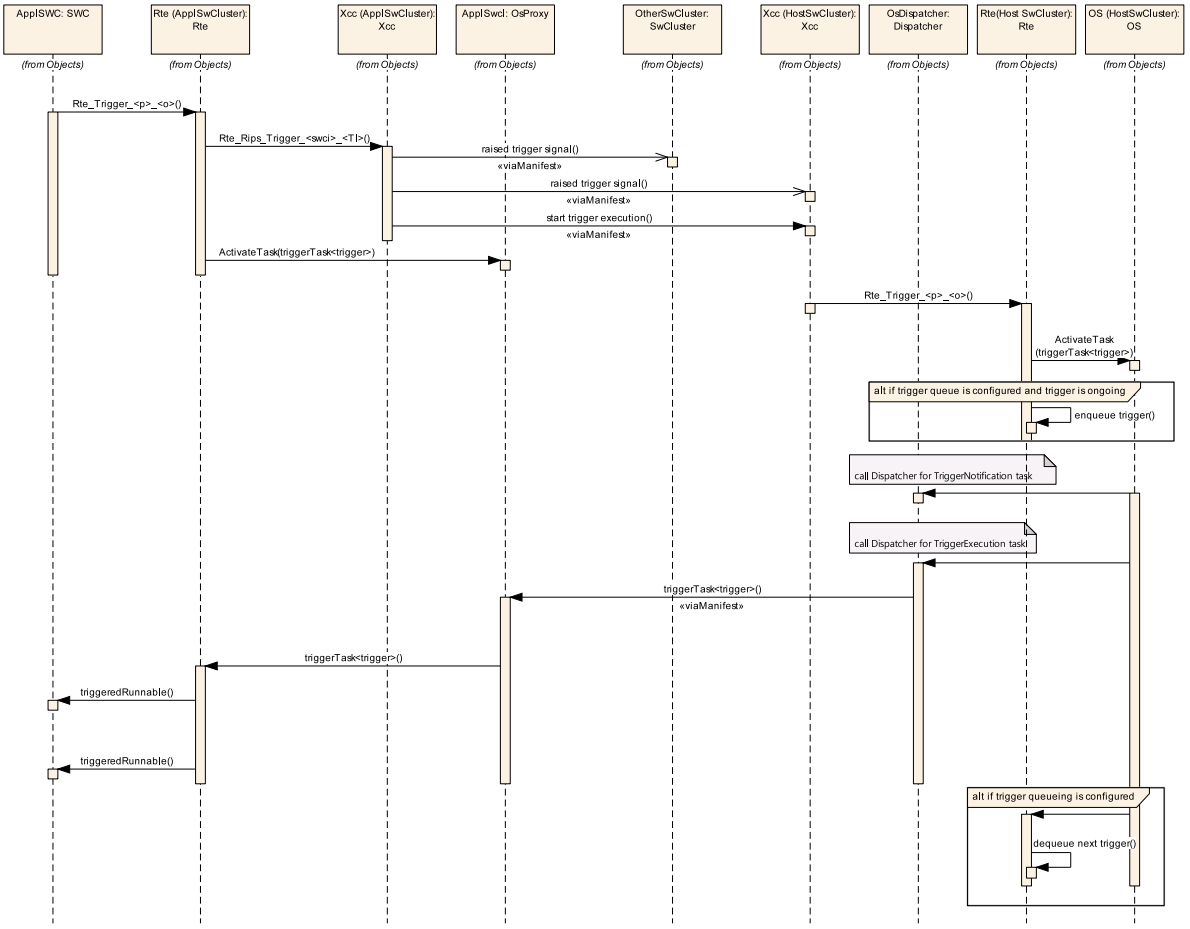

Figure 7.6: Execution of a trigger provided by an Application Software Cluster

- The `trigger source` calls the `Rte_Trigger` API, to raise the trigger.
- The `Cross Cluster Communication` gets notified, via the call of the `Rte_Rips_Trigger` Service.
- The `Cross Cluster Communication` in the `Application Software Cluster` informs the `Cross Cluster Communication` in the `Host Software Cluster` that a trigger was raised. This notification is a `Cross Cluster Communication` internal interface, which is not standardized.
- In the `Host Software Cluster`, the `Cross Cluster Communication` raises the trigger in the RTE, via `Rte_Trigger` API. In case of a configured trigger queue, this would enqueue the trigger.
- After the start of the trigger execution on the `Host Software Cluster`, the according tasks are scheduled (potentially on multiple partitions). This in turn schedules, via `Dispatchers`, the `Application Software Cluster` local proxy tasks.
- In case of a configured trigger queue, the dequeue operation is executed after the last `triggered ExecutableEntitys` terminated.

The figure 7.7 illustrates the principle sequence, when a trigger is raised by a Host Software Cluster.

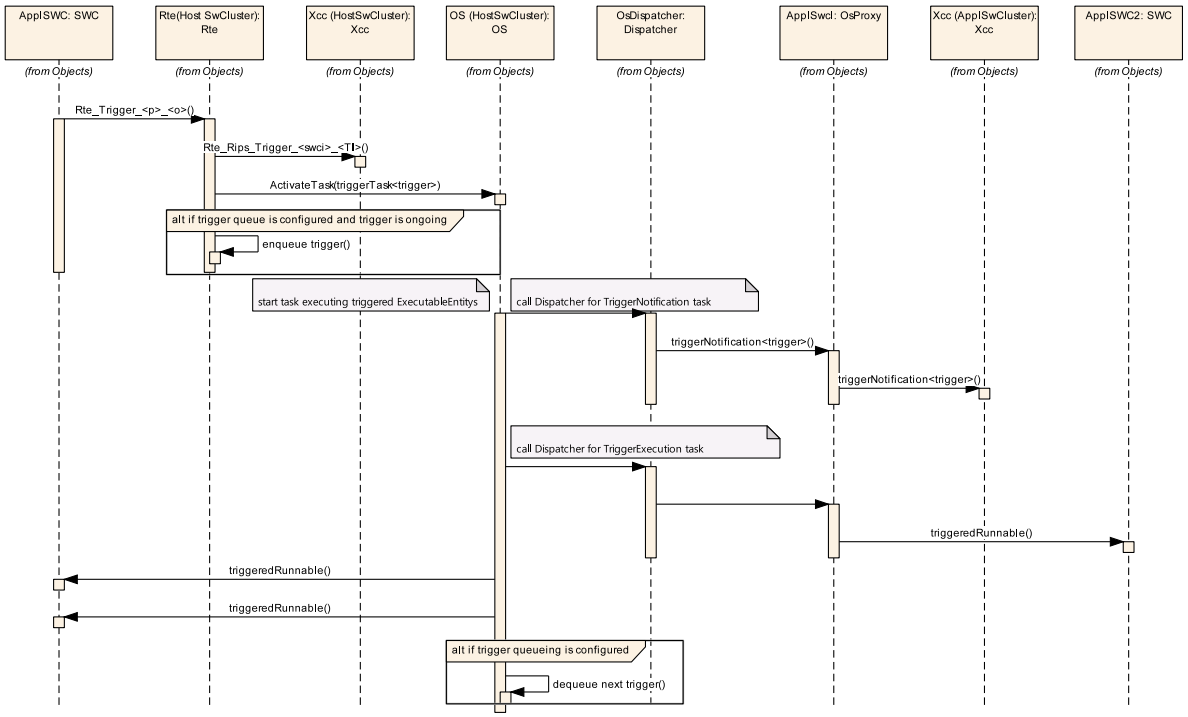

Figure 7.7: Execution of a trigger provided by an Host Software Cluster

- The trigger source calls the Rte_Trigger API, to raise the trigger.
- The Cross Cluster Communication gets notified, via the call of the Rte_Rips_Trigger Service.
- The RTE activates the tasks used for execution of triggered Executable-Entitys. If a queue is configured, and a trigger execution is already ongoing, the RTE in the Host Software Cluster enqueues the trigger.
- After the start of the trigger execution on the Host Software Cluster, the according tasks are scheduled (potentially on multiple partitions). This in turn, schedules, via Dispatchers, the Application Software Cluster local proxy tasks.
- In case of a configured trigger queue, the dequeue operation is executed after the last triggered ExecutableEntitys terminated.

These two principle sequences lead to the following requirements on the Cross Cluster Communication, in case a Software Cluster provides a mode.

[SWS_SwCluC_03055]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication shall signal to other Software Clusters that a trigger was raised.⌋

Note: This functionality can be implemented inside the Rte_Rips_Trigger service. Function wise, it is up to the implementation, if the [SWS_SwCluC_03055] is already fulfilled via the connection to the Host Software Cluster.

[SWS_SwCluC_03056]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication shall, at the Software Cluster providing the trigger, add a Provide Resource Entry and, at Software Clusters requiring the trigger, an according Require Resource Entry in the Binary Manifest.⌋

[SWS_SwCluC_03056] ensures that a missing trigger source in a clustered ECU gets detected, even if the implementation may rely on the according Dispatchers.

[SWS_SwCluC_03058]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication of the Host Software Cluster shall implement a trigger source for each trigger, which needs to be provided by an Application Software Cluster.⌋

With this trigger source, it is possible to schedule the according Dispatchers for trigger notification, and Dispatchers for Software Cluster local trigger execution tasks.

To ease the realization of [SWS_SwCluC_03058], all triggers provided by Application Software Clusters, shall be consumed by the Host Software Cluster.

[SWS_SwCluC_CONSTR_03059]

Upstream requirements: SRS_SwCluC_00105, SRS_BSW_00167

⌈The Ecu Extract of the Host Software Cluster shall, for each trigger provided by an Application Software Cluster, own a required trigger port at the CompositionSwComponentType of the rootSoftwareComposition.⌋

[SWS_SwCluC_03060]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication of the Host Software Cluster shall raise the triggers signaled by Application Software Clusters, via the related trigger source. ([SWS_SwCluC_03058])⌋

Note: In case the Host Software Cluster provides the trigger, the trigger source is already part of the software components or BSW Modules belonging to the Host Software Cluster, which are directly interacting with the RTE or SchM.

[SWS_SwCluC_03061]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication of the Host Software Cluster shall notify the Application Software Cluster, when the execution of a trigger tasks starts.⌋

Note: This notification is used by the Application Software Cluster to schedule the trigger notification runnable, which raises the trigger at the cluster local RTE.

[SWS_SwCluC_03062]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide a Complex Driver Software Component on each EcucPartition, where a trigger source according to [SWS_SwCluC_03058] is configured. This component is later called trigger proxy component.⌋

7.3.7.3 Host Software Cluster requiring a trigger

As explained in section 7.3.7.2, the Host Software Cluster is already aware about an existing trigger communicated cross Software Clusters. Therefore, it technically already receives the triggers raised by Application Software Clusters. In case the Host Software Cluster owns software components that require the the trigger as well, it shall connect them to its already existing trigger source, according to [SWS_SwCluC_03058].

[SWS_SwCluC_03063]

Upstream requirements: SRS_SwCluC_00104

⌈The Cross Cluster Communication shall provide the AssemblySwConnectors between the provided trigger ports of the trigger proxy component, and the required trigger ports at the software components requiring the trigger, if the trigger is required by the Host Software Cluster.⌋

7.3.7.4 Application Software Cluster requiring a trigger

For each required trigger, the Application Software Cluster defines OsTask(s), for the trigger notification.

When the trigger gets executed by the Host Software Cluster, the Cross Cluster Communication raises trigger at the RTE of the Application Software Cluster, with the regular Rte_Trigger API related to a trigger port.

Note: The RTE in the Application Software Cluster shall still call ActivateTask(s), to trigger the task(s) used for the execution of triggered ExecutableEntitys. This call can still be used by the Host Software Cluster to trigger the task executions in the Host Software Cluster, if needed.

For this purpose, the Cross Cluster Communication defines a RunnableEntity with an OsTaskExecutionEvent, for each trigger and on each partition, a trigger notification is configured. When the trigger is executed, the Host Software Cluster schedules the task(s), which in turn schedule the 'proxy' task(s) in the Application Software Cluster.

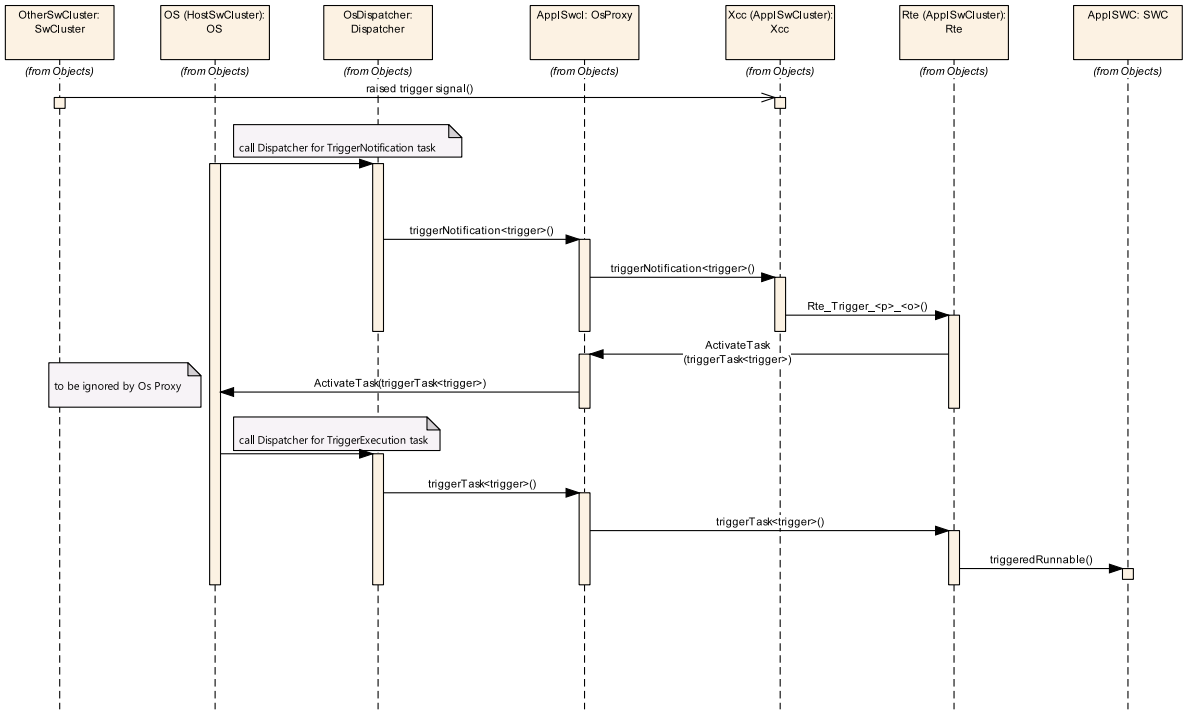

Figure 7.8: Execution of a trigger in an Application Software Cluster

[SWS_SwCluC_03048]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication shall provide a Complex Driver Software Component, on each EcucPartition where a trigger port needs to be provided. This component is later called trigger proxy component.⌋

[SWS_SwCluC_03049]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication shall provide a provided trigger port, on each trigger proxy component.⌋

[SWS_SwCluC_03050]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication shall provide the AssemblySwConnectors between the provided trigger ports and the required trigger ports, at the software components requiring the trigger.⌋

[SWS_SwCluC_03051]

Upstream requirements: SRS_SwCluC_00105

⌈The Cross Cluster Communication shall provide a trigger notification runnable with an OsTaskExecutionEvent, for each trigger on each partition a trigger notification is configured. The RunnableEntity shall define a ExternalTriggeringPoint.⌋

[SWS_SwCluC_03052]

Upstream requirements: SRS_SwCluC_00105

⌈When the trigger notification runnable is executed, it shall raise the trigger via Rte_Trigger to the RTE.⌋

[SWS_SwCluC_CONSTR_03053]

Upstream requirements: SRS_SwCluC_00105, SRS_BSW_00167

⌈For each EcucPartition on which the trigger notification is configured for a trigger required by an Application Software Cluster, a OsTask, with a related SwCluCOsProxyOsTask shall be configured. On the Host Software Cluster, a SwCluCOsProxyOsTaskDispatcher shall be configured.⌋

[SWS_SwCluC_CONSTR_03054]

Upstream requirements: SRS_SwCluC_00105, SRS_BSW_00167

⌈For each required trigger and EcucPartition on which triggered ExecutableEntitys needs to be executed, at least one OsTask, with a related SwCluCOsProxyOsTask shall be configured. On the Host Software Cluster, a SwCluCOsProxyOsTaskDispatcher shall be configured.⌋

7.3.8 Parameter Communication

[SWS_SwCluC_03006]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00106

⌈The Cross Cluster Communication of the Software Cluster Connection shall provide one constant default parameter instance for each assigned Parameter Communication Graph, which contains NO PPortPrototype.⌋

[SWS_SwCluC_03007]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00106

⌈The default value instance shall be mapped to a CALPRM memory section, according to document [15].⌋

Please note as well section 7.3.2.4, which is relevant for calibration parameters instantiated by Software Cluster Connection.

[SWS_SwCluC_03008]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00106

⌈The default value instance shall be initialized according to the ParameterRequireComSpec.initValue, in the RPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract.⌋

[SWS_SwCluC_03009]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00106

⌈If the corresponding resource in the Binary Manifest is connected, the Rte_Rips_Prm shall return the value of the connected parameter.⌋

[SWS_SwCluC_03010]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00106, SRS_SwCluC_00213

⌈If the corresponding resource in the Binary Manifest is NOT connected, the Rte_Rips_Prm shall return the value of the default parameter instance of [SWS_SwCluC_03006].⌋

Note: The behavior of Rte_Rips_Prm function, according to [SWS_SwCluC_03009] and [SWS_SwCluC_03010], can be implemented by referencing the default parameter instance as the default handle in the Binary Manifest.

[SWS_SwCluC_03011]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00106

⌈For each assigned Parameter Communication Graph, which contains the PPortPrototype, the Cross Cluster Communication of the Software Cluster Connection shall make the parameter instance, according to [SWS_Rte_80130], accessible for other Software Clusters, via a Resource Entry in the Binary Manifest.⌋

7.3.9 Error Classification

Together with the RTE, the Cross Cluster Communication of the Software Cluster Connection implements the Run-Time Environment of the Software Cluster. The usual development errors checking API parameters are already covered by the development error detection of RTE. In addition, communication infrastructure errors are not reported as Production Errors. Therefore, the following sections do not define error codes, and are marked as not applicable.

7.3.9.1 Development Errors

Development errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.3.9.2 Runtime Errors

Runtime errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.3.9.3 Production Errors

Production Errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.3.10 Initialization of Xcc

Please note, that the Cross Cluster Communication offers two initialization functions for each Software Cluster. This enables that each Software Cluster can first initialize its memories which are intended to be read by other Software Clusters before initializations are executed which depend on that to be read memories. To ensure proper initialization behavior all SwCluC_Xcc_Init1 have to be called before the SwCluC_Xcc_Init2 functions.

[SWS_SwCluC_CONSTR_03404] SwCluC Xcc initialization order

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00108

⌈All SwCluC_Xcc_Init1 functions of any Software Cluster on the machine shall be called before the call of the SwCluC_Xcc_Init2 functions of any Software Cluster on the same machine.⌋

7.4 Proxy Modules

7.4.1 Overview

Since an Application Software Cluster does not contain all BSW modules (in extreme case no BSW modules), those missing APIs, as well as the EcuC configuration elements, need to be substituted.

In the AUTOSAR Layered Software Architecture, the dependency between BSW modules can be generalized as follows:

A higher layer BSW module uses the APIs of a lower layer BSW module. In the opposite direction, the lower layer BSW module may call callback functions of the higher layer BSW module. In case of AUTOSAR Interfaces, additional software component descriptions are required to enable the RTE generation. If the interface between those BSW modules is configurable, the ECU configuration provides the information about symbolic name values and ID values.

Of course, the AUTOSAR Layered Software Architecture also has horizontal dependencies. In this case, rotate the previous paragraph by 90 degrees.

It is important to state that the different BSW Module are directly using the C-interfaces of each other, without abstraction (as the RTE provides between SWCs). This, in turn, causes strong implementation dependencies, like fixed names of include files and C-functions.

Since the actual implementation and integration of the Classic Platform architecture is now split into Software Clusters, missing BSW Modules - those to which interfaces are existing - need to be substituted. The Figure 7.9 illustrates the proxy module approach, for a horizontal interface dependency.

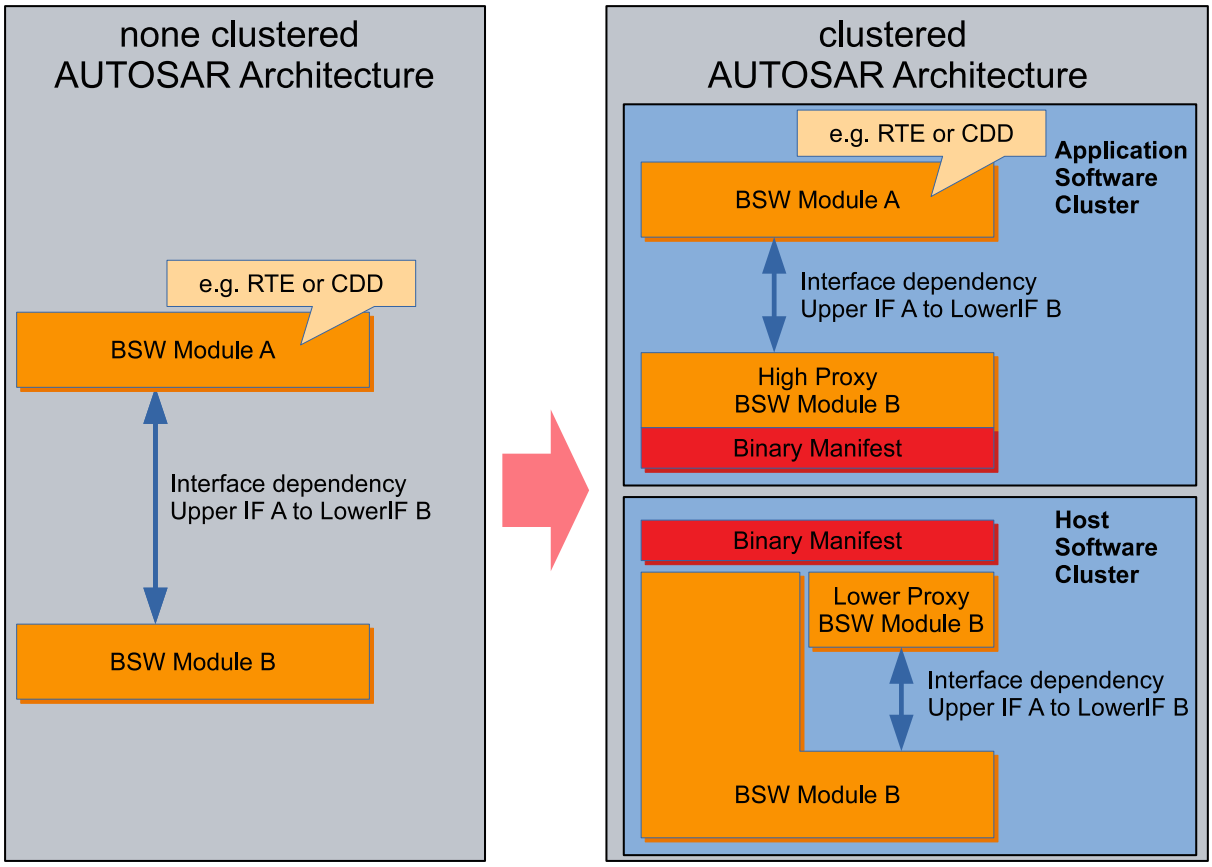

Figure 7.9: Proxy module approach

The C-interfaces of the lower layer BSW module (Module B), towards the higher layer BSW module (Module A), are provided by the High Proxy Module B, in the Software Cluster SWCL 1. The C-interfaces, ECU-C, and Component Descriptions of Module B are provided in the same name space / ARPackage structure as the original Module B. For the users of Module B (in this example Module A), the substitution of Module B by the High Proxy Module B is transparent. Usually, the offered API functions are almost without logic, and are just redirecting the calls towards the substituted module.

In the `Host Software Cluster`, the Low Proxy Module acts as a substitute for all 'user modules' in other Software Clusters. It implements the callback functions, which are normally configured in Module B.

The High Proxy Module and Low Proxy Module are connected via the means of the `Binary Manifest`. Therefore, the functions of the High and Low Proxy modules may contain functionality, which handles this `Binary Manifest` link and the situations if the link is missing. For instance, returning a reasonable error code or even providing some replacement value.

In addition, the High Proxy Module can implement functionality to localize configuration decisions. For instance, in order to support a SWCL local growing of the NV block over integration steps, it could be possible to adjust the locally used size of a NV block. Of course, this is only possible if the configured size in the real NvM is sufficiently large.

By this approach, the SWCs and BSW modules inside the Application Software Clusters call the common BSW inside the Host Software Cluster via proxies. When doing so, the partitioning of the ECU SW has to be considered, as described in section 7.4.2.2 Partitions.

This section specifies the general Proxy Module pattern. Additionally, requirements for specific Proxy Modules are in section 7.4.3. Additional Proxy Modules might be standardized by AUTOSAR in future releases.

7.4.2 Abstract Proxy Module Pattern

This section provides some generic design principles and common requirements, relevant for Proxy Module implementations.

Basically, a High Proxy Module provides a kind of facade, which hides the Software Cluster Connection specific mechanisms to access the real BSW Module. In the AUTOSAR Architecture, such a facade has to hide dependencies to

- Standardized Interfaces
- Standardized AUTOSAR Interfaces
  and the
- ECU Configuration

In addition to the essential need to provide the interfaces to the BSW Module user in the Application Software Cluster, the Proxy Module should implement abstraction functionalities. Those abstraction functionalities decouple the needed configuration in the High Proxy Module, from the configuration done in the BSW Module of the Host Software Cluster. The introduction of abstraction functionalities is a case-by-case decision. One the one hand, they might need a functional support of the according BSW Module, or might lead to functional restrictions. On the other hand, such abstractions are beneficial, since in some cases they can be used to avoid a reconfiguration of the Host Software Cluster, after change in the Application Software Cluster.

Please note that the following requirements only have limited applicability to an OS Proxy, since the OS APIs are already designed to be called from all the partitions.

7.4.2.1 General Proxy functionality

[SWS_SwCluC_02000]

Upstream requirements: SRS_SwCluC_00203

⌈The Proxy Module shall support that id values, required to access the BSW service API of the Host Software Cluster, can change, without reconfiguration or rebuild of the Application Software Cluster using the BSW service.⌋

7.4.2.2 Partitions

Partitions are shared between the Host Software Cluster and Application Software Clusters. This also applies to the properties of the related OsApplications and OsTasks. Consequently, Application Software Clusters are not strictly separated from the Host Software Cluster. Due to performance reasons, it is possible to share the identical partiton of the Host Software Cluster with many Application Software Clusters. If Application Software Clusters share the same partition, they are also not strictly separated from each other. In contrast, if a Software Cluster has multiple partitions (e.g. on different safety levels), those partitions are separated from each other.

When a partition crossing function call is required, usually some close interaction with the Operating System is necessary - especially in case of different safety levels. However, the Operating System is not directly available inside an Application Software Cluster. Additionally, it needs to be considered that passing synchronous function calls cross partitions, may have an impact on the scheduling behavior of the software. Therefore, it is preferable to do this in the Host Software Cluster, or - even better - to avoid this by implementing the master satellite pattern in the related BSW modules.

As a consequence:

Inside a partition, such direct function calls to the BSW module in the Host Software Cluster are easily possible, with little overhead. If a BSW service is needed in a specific partition (including safety partition), the Host Software Cluster should offer the service interface in this partition. If this is not possible, the according transition shall be implemented in the Host Software Cluster. To support this, the Proxy Modules have to grant a partition write access to their interfaces.

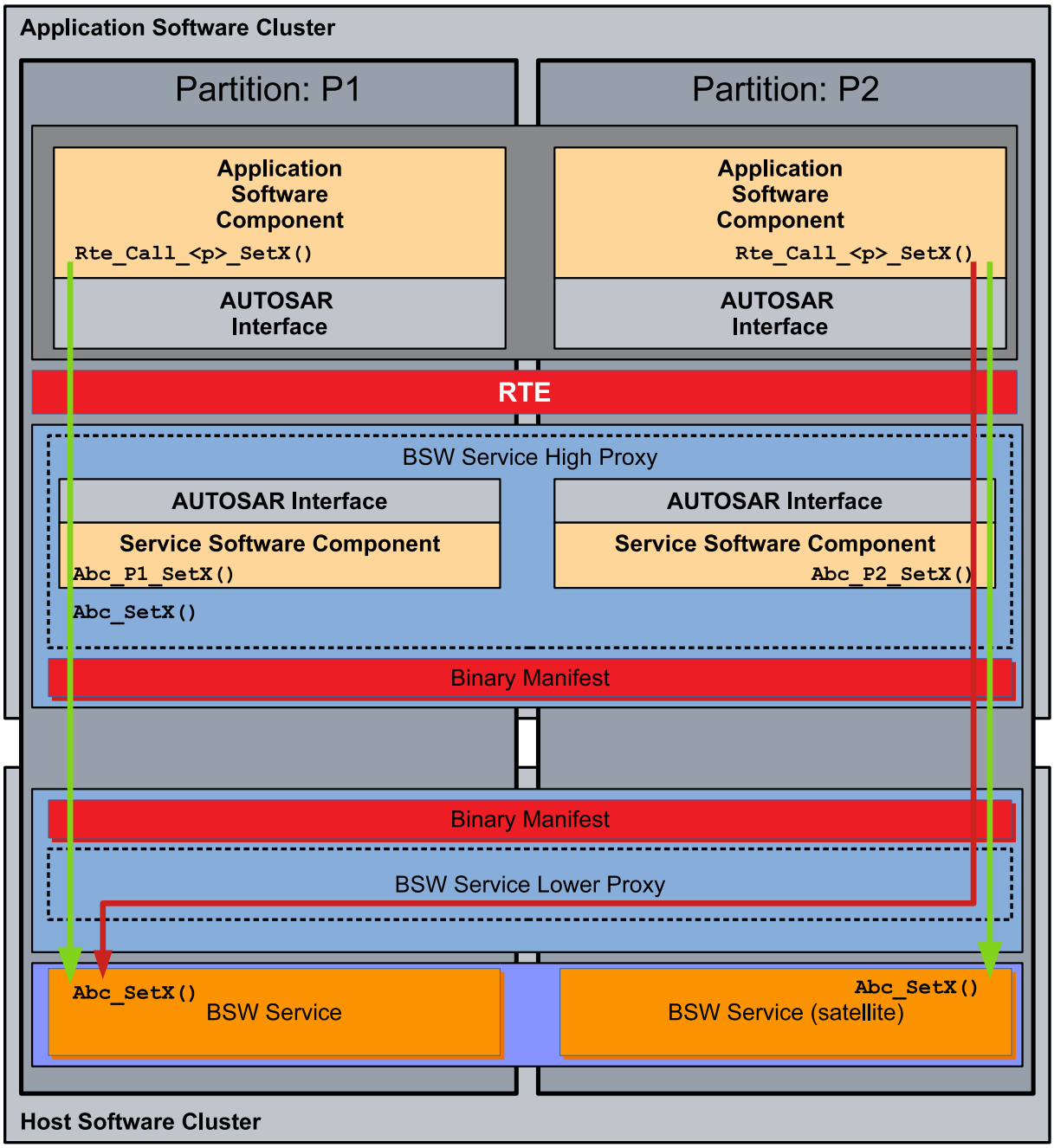

Figure 7.10: Proxy Modules and Partitions

Figure 7.10 illustrates this approach. The `AUTOSAR Interfaces` are offered in each partition, via a dedicated `Service Software Component`. The related `Standardized AUTOSAR Interface` is provided in exactly one Partition, in which the BSW module integration is supported.

A call from the Application Software Component is passed vertically through the partition, down to the BSW Module or its satellite (green arrow).

If such a satellite is not available, the Low Proxy Module can implement (e.g. via `SchM`) a cross partition call (red arrow).

[SWS_SwCluC_02001] Partition specific APIs

Upstream requirements: SRS_SwCluC_00202

⌈For BSW modules not supporting distribution to multiple partitions, the High Proxy shall provide the applicable Standardized AUTOSAR Interfaces once per configured EcucPartition. In doing so, the APIs are provided with the orignal Mip, at the EcucPartition where the SwCluCNativeBswApi is set to true. For all the other BSW API instances, the Mip is replaced by <Mip>_<shortName of EcucPartition>. Otherwise, if the BSW module does support BSW module APIs supporting multiple partions at once (e.g global scope API according [18] Chapter 2 *"BSW Distribution in Multi-Core Systems"*), APIs are provided once by the High Proxy for all configured EcucPartitions, as implemented in the original BSW module.⌋

Note: For instance Com, ComM and NvM are already supporting BSW Multicore Distribution. Hence the specification of SwCluC will not require any partition specific API treatments in addition.

[SWS_SwCluC_02002]

Upstream requirements: SRS_SwCluC_00201

⌈The High Proxy shall create one partition specific Service Software Component per configured EcucPartition, with the name <Mip>_<EcucPartition shortName>.⌋

[SWS_SwCluC_02003]

Upstream requirements: SRS_SwCluC_00201

⌈The High Proxy shall provide the Ports belonging to a specific CpSoftwareClusterServiceResource, at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped to.⌋

7.4.2.3 Unconnected Service Resources

[SWS_SwCluC_02004]

Upstream requirements: SRS_SwCluC_00213

⌈The High Proxy shall implement a 'reasonable' behavior for a service resource, if the connection to the Resource Provider does not exist. In any case, the OUT arguments of functions shall return a 'neutral' value, or the value which can be assumed after a reset. The functional behavior should be like the state directly after a reset, before further activity occured.⌋

Note: To decide what is 'reasonable', a functional understanding of the service resource is required. Therefore, [SWS_SwCluC_02004] can only give some rough expectations.

7.4.2.4 Ecu Configuration Principles

[SWS_SwCluC_02005]

Upstream requirements: SRS_BSW_00159

⌈To resolve the ECU Configuration dependency, the Proxy Module implementation shall define a Vendor Specific Module Definition [16], where the ECU Configuration container and parameters are used to configure the interface towards the user of the substituted BSW Module.⌋

Typically, those are the containers and parameters configuring the variation (e.g. existence) of `Standardized AUTOSAR Interfaces`, or parameters relevant for `symbolic name value`s.

In addition, these containers need to have a relationship to the software cluster resource pool, in order to manage the configuration needs exchange between the different `Software Cluster` providers. Furthermore, the `CpSoftwareClusterServiceResource` element defines the `globalResourceId`, which is required for the `Resource Entrys` in the `Binary Manifest`.

[SWS_SwCluC_02006]

Upstream requirements: SRS_BSW_00159

⌈The Proxy Module implementation shall define an `EcucContainerDef`, which defines a mapping between

- the `EcucContainerDef`(s) identifying the module user channel in the `StMD` of substituted BSW Module

  AND

- the `CpSoftwareClusterServiceResource` representing the user channel in the software cluster resource pool. The `EcucForeignReferenceDef` shall be named `SwCluCResourceRef`.

⌋

Note that this `EcucContainerDef` of [SWS_SwCluC_02006] can also be used to add additional Parameters, to control the connection behavior between `High Proxy` and `Low Proxy`. For an example, see: `SwCluCNvMProxyNvBlock`.

[SWS_SwCluC_02007]

Upstream requirements: SRS_BSW_00159, SRS_SwCluC_00204

⌈The Proxy Module implementation shall define, for each different Proxy Module, an `EcucEnumerationParamDef` named `SwCluCProxyGeneration<Mip>`, with the `literal`s

- `PROXY_DISABLED`, to disable the according Proxy Module code and AUTOSAR model generation

- HIGH_PROXY, to enable the according High Proxy Module code and AUTOSAR model generation
- LOW_PROXY, to enable the according Low Proxy Module code and AUTOSAR model generation

⌋

7.4.2.5 Proxy Modules and Binary Manifest

It is the responsibility of the Proxy Module implementation to put its Resource Entrys into the Binary Manifest of the respective Software Cluster. This includes service user specific channels (e.g. for a specific NV block), or some general links - typically named Base Socket - to link some generic infrastructure between Low and High Proxy (e.g. the set of BSW APIs).

[SWS_SwCluC_02008]

Upstream requirements: SRS_SwCluC_00212

⌈The Proxy Module implementation of the Software Cluster Connection shall put one Resource Entry, for each associated CpSoftwareClusterServiceResource, into the Binary Manifest.⌋

7.4.2.6 Proxy Modules and Base Sockets

Proxy Modules may define so called Base Sockets to implement some general connections to generic infrastructure between Low Proxy Module and High Proxy Module (e.g. the set of channel independent BSW APIs). Those proxies define a parameter SwCluCNoOfSupportedApplClusterConnections configuring how many Application Software Clusters can share a common socket provided by the Host Software Cluster.

[SWS_SwCluC_02009] Default behavior of SwCluCNoOfSupportedApplClusterConnections

Upstream requirements: SRS_SwCluC_00212

⌈If the parameter SwCluCNoOfSupportedApplClusterConnections is not given in the configuration of Low Proxy Module's Base Socket the according Base Socket shall support at most one connection to an Application Software Cluster.⌋

For example, the parameter SwCluCNoOfSupportedApplClusterConnections is defined for the NvM Low Proxy in the container SwCluCNvMBaseSocket.

7.4.3 Specific Proxy Module Requirements

7.4.3.1 OS Proxy

7.4.3.1.1 Enable OS Proxy Generation

[SWS_SwCluC_02200]

Upstream requirements: SRS_SwCluC_00204

⌈The `Os High Proxy` code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter `SwCluCProxyGenerationOs` is set to `HIGH_PROXY`.⌋

[SWS_SwCluC_02201]

Upstream requirements: SRS_SwCluC_00204

⌈The `Os Low Proxy` code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter `SwCluCProxyGenerationOs` is set to `LOW_PROXY`.⌋

7.4.3.1.2 Overview OS Proxy functionality

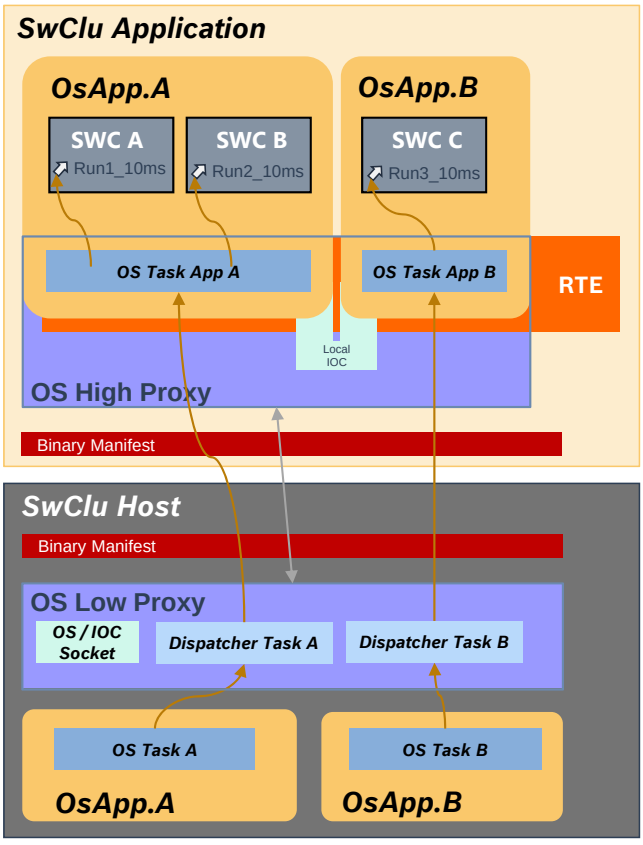

Figure 7.11: Overview of Os Proxy

As explained in chapter 7.1.1 and 7.4.1, the concept of Software Clusters also has to provide a way to divide some parts of an AUTOSAR Os into smaller units (see document [5] as reference for Os). Basically, any RunnableEntity of a Software Cluster shall be executed on a target machine. Since the system is divided into a machine dependent part (Host Software Cluster), and several machine independent parts (Application Software Clusters), an interface layer in between is required. This layer will be realized using the proxy concept.

The Application Software Cluster contains proxies for those parts of the OS implementation that are needed to complete the runtime environment (RTE). This is called Os High Proxy.

The Host Software Cluster will implement the AUTOSAR Os, as well as the overall configuration of the machine. The Os Low Proxy makes this AUTOSAR Os accessible by the Os High Proxy, via the Binary Manifest. Both are linked together via the information held in the Binary Manifest, which enables an API exchange between Software Clusters.

The OS features (e.g. tasks, resources, spinlocks) supported by the Os High Proxy are configured with EcucModuleConfigurationValues for an Os EcucModuleDef, with according OsTask, OsResource, OsSpinlock, etc. containers. In the Application Software Cluster and in the Host Software Cluster, the configuration of SwCluCOsProxyOsTask, SwCluCOsProxyOsResource, SwCluCOsProxyOsSpinlock, etc. containers map the OS configuration containers to CpSoftwareClusterServiceResources, defining each the globalResourceId and the isMandatory attribute, relevant for the corresponding Resource Entry in the Binary Manifest.

The Os Low Proxy provides an Os- and IOC-socket for each Application Software Cluster. It also has a Dispatcher, which is responsible for calling each Os-Task of an Application Software Cluster. The Os- and IOC-socket provides basic hooks to the Application Software Cluster. These are used, for example, to do basic initialization of local Os and IOC channels, to forward general APIs like spinlock, or to do a context switch.

The Dispatcher Runnable controls and calls the Os-Tasks from the Application Software Cluster. It is only configured in the Os Low Proxy and not present in the Os High Proxy. It is linked to an RTEEvent that is scheduled in the Host. The Runnable calls the Os High Proxy Task via the entry found in the Binary Manifest.

The Host Software Cluster provides the implementation of OS Applications, and an abstraction of them in the Os Low Proxy. The Os-Application context inside an Application Software Cluster matches to the corresponding OS Application in the Host Software Cluster. The Os High Proxy task is executed in the same context as the low proxy task, and has the same level of priority, trust and access.

As mentioned before, a connection between Os Low Proxy and Os High Proxy is required. It will be established with the mechanism described in chapter 7.1.5.2 (Connecting Resources). In this chapter, the metaclasses for Os service resources are defined: CpSoftwareClusterServiceResources. Essential to run an Application Software Cluster are resources for the Os- and IOC-socket and the Os-ProxyTasks. These resources are part of the system's resource pool (CpSoftwareClusterResourcePool), which includes the description of their relevant attributes and a unique globalResourceId. (CpSoftwareClusterResource.globalResourceId). A valid connection can only be established, if all attributes configured for those resources match during the connection process.

To achieve a robust system, and to avoid erroneous behavior, it is recommended to mark Os service resources as mandatory.

Detailed requirements can be found below.

7.4.3.1.3 General requirements

[SWS_SwCluC_02202]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Proxy shall support the execution of OS Task from an Application Software Cluster, in the context of the Host Software Cluster.⌋

[SWS_SwCluC_CONSTR_02203] Constraint for Interrupts

Upstream requirements: SRS_SwCluC_00214

⌈Interrupts are not supported for an Application Software Cluster, since it does not have access to the hardware or peripherials.⌋

[SWS_SwCluC_02204]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Proxy shall support a socket mechanism to hook several Application Software Clusters onto the Host Software Cluster.⌋

For example, this can be used to support Application Software Cluster local IOC communication paths.

[SWS_SwCluC_02205]

Upstream requirements: SRS_SwCluC_00214

⌈The Os High Proxy interfaces shall be described with the parameters, and according containers, as defined in Os EcucModuleDef.⌋

[SWS_SwCluC_02206]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Proxy` interfaces shall be configured as a service resource, with the parameters as defined in the `SwCluCOsProxy` container.⌋

7.4.3.1.4 OS High Proxy

[SWS_SwCluC_02210]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os High Proxy` shall provide the header file `Os.h` and `Os_Cfg.h` .⌋

[SWS_SwCluC_02211]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os High Proxy` shall provide a proxy `OS Task`, to which events from the `Application Software Cluster` can be mapped, during the design of an `Application Software Cluster`.⌋

[SWS_SwCluC_02230]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os High Proxy` shall support the implementation of a task body of the proxy `OS Task`, with the `TASK()` macro.⌋

[SWS_SwCluC_02212]

Upstream requirements: SRS_SwCluC_00202, SRS_SwCluC_00214

⌈The `Os High Proxy` shall provide a set of OS APIs that are required by the `Application Software Cluster`:

- `ActivateTask`
- `ChainTask`
- `TerminateTask`
- `Schedule`

⌋

[SWS_SwCluC_02213]

Upstream requirements: SRS_SwCluC_00214

⌈If `SwCluCOsProxyOsTask.SwCluCOsProxyTaskActivation.PASS_TASK_ACTIVATION` is set for a specific `OsTask`, the API `ActivateTask` called for this `OS Task` shall trigger a task activation in the `Host Software Cluster`.⌋

[SWS_SwCluC_02229]

Upstream requirements: SRS_SwCluC_00214

⌈If SwCluCOsProxyOsTask.SwCluCOsProxyTaskActivation.OMIT_TASK_ACTIVATION is set for a specific OsTask, the API ActivateTask called for this OS Task returns without effect.⌋

[SWS_SwCluC_02214]

Upstream requirements: SRS_SwCluC_00214

⌈In an Application Software Cluster, the Os High Proxy shall provide a 'ChainTask' API which is implemented as macro and must contain a return statement.⌋

Please note: The OS Task chaining can only be handled in the OS implementation in the Host Software Cluster.

[SWS_SwCluC_CONSTR_02282] Constraint for ChainTask

Upstream requirements: SRS_SwCluC_00214

⌈In an Application Software Cluster, the 'ChainTask' API shall only be called at the end of the function body of an 'OS Task'.⌋

[SWS_SwCluC_02215]

Upstream requirements: SRS_SwCluC_00214

⌈In an Application Software Cluster, the Os High Proxy shall provide a 'TerminateTask' API which is implemented as a macro and must contain a return statement.⌋

Please note: The real OS Task termination can only be handled in the OS implementation in the Host Software Cluster.

[SWS_SwCluC_CONSTR_02283] Constraint for TerminateTask

Upstream requirements: SRS_SwCluC_00214

⌈In an Application Software Cluster, the 'TerminateTask' API shall only be called at the end of the function body of an 'OS Task'.⌋

Rationale:

When developing an Application Software Cluster, the Os High Proxy has no ability to call to the 'real' Os to activate or terminate tasks based on calls to 'ChainTask', 'ActivateTask' or 'TerminateTask'. However, this does not mean that calls to these functions cannot exist within application cluster code. It simply means that a SwCluC module implementation must contain 'stub' representations of these functions in an Os High Proxy which have no effect when called within an application cluster which has not been integrated in the final system which includes a Host Software Cluster.

However, given that application clusters will eventually be integrated together with a host cluster, code in an application cluster must follow some rules to ensure that their usage of task management APIs do not clash with the intended use case of cyclic execution in cluster based configurations.

Note: Where constraints [SWS_SwCluC_CONSTR_02282] and [SWS_SwCluC_CONSTR_02283] restrict the usage of `ChainTask` and `TerminateTask` in an `Application Software Cluster`, no such constraint exists to restrict the usage of 'ActivateTask'.

In a clustered software architecture, the dynamic behavior of the ECU is controlled by the `Host Software Cluster`. Hence, the `Host Software Cluster` is in charge to control the activation of `OS Tasks`.

Nevertheless, use cases may exist to sporadically activate `OS Tasks` on demand of an `Application Software Cluster`. For this reason, the ActivateTask API is offered by the `Os High Proxy` which is fully functional in case of [SWS_SwCluC_02213]. In opposite to 'TerminateTask' and 'ChainTask' the 'ActivateTask' API usage is not bound to the task body. The usage of 'ActivateTask' APIs in `Application Software Cluster` needs to be carefully aligned with the overall dynamic scheduling concept of the ECU!.

[SWS_SwCluC_02216]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os High Proxy` shall provide default handles for resources, in case a connection to the host system could not be established successfully between `Application Software Cluster` and the `Host Software Cluster`. The default implementation shall be a neutral behavior, with the return value `E_OK`, to support "standard status" from OS.⌋

See the section 7.4.3.1.5 'Os Low Proxy', for an explanation of how a missing connection is handled in the `Host Software Cluster`.

[SWS_SwCluC_02217]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os High Proxy` shall have a service resource `SwCluCOsProxyOsTask`, for each OsTask required in the `Application Software Cluster`. For this resource, an entry in the `Binary Manifest` shall be created, to be able to link the task to its OsLowProxy implementation in the `Host Software Cluster`. Since the content of the OsTask is created by the RTE, this API shall be used and registered as service resource. It is recommended that this resource is marked with the mandatory flag (`SwCluCBManifIsMandatory`).⌋

Note that the `SwCluCOsProxyOsTaskDispatcher` is only required in the `Host Software Cluster` / `Os Low Proxy` and should not be configured in the `Application Software Cluster` / `Os High Proxy`.

[SWS_SwCluC_02219]

Upstream requirements: SRS_SwCluC_00214

⌈The Os High Proxy shall contain a required service resource SwCluCOsProxyFunctionDispatcher. This function dispatcher is used to call functions in context of the Host Software Cluster. For this resource, an entry in the Binary Manifest shall be created, to be able to link it to the OsLowProxy dispatcher in the Host Software Cluster.⌋

For example, to call an Application Software Cluster local Bsw-Module Init function, in dedicated context of the Host Software Cluster.

[SWS_SwCluC_02220]

Upstream requirements: SRS_SwCluC_00214

⌈The Os High Proxy shall contain a required service resource SwCluCOsProxyOsBaseSocket. This collects all hooks needed to be linked to the Host Software Cluster. It also contains Application Software Cluster local Os specific functions for StartOs, initialize and shutdown. It should also cover the local IOC, if present. It is recommended that this resource is marked with the mandatory flag (SwCluCBManifIsMandatory).⌋

Please note: This is highly vendor specific.

[SWS_SwCluC_02221]

Upstream requirements: SRS_SwCluC_00214

⌈Since an Application Software Cluster can contain several OS Applications, the Os High Proxy shall be able to provide a local IOC implementation for the cluster internal communication paths between OS Applications. The link to the Host Software Cluster shall be established as part of the SwCluCOsProxyOsBaseSocket service resource.⌋

[SWS_SwCluC_02222]

Upstream requirements: SRS_SwCluC_00214

⌈The Os High Proxy shall require a service resource SwCluCOsProxyOsApplication, which is used in the Application Software Cluster, and needs to be matched to the Host Software Cluster.⌋

This service resource represents the actual configuration of the Application Software Cluster's local OS Application, with its parameters and linked local OS objects.

[SWS_SwCluC_02223]

Upstream requirements: SRS_SwCluC_00214

⌈The Os High Proxy shall require a service resource SwCluCOsProxyOsResource, which is used in the Application Software Cluster, and needs to be matched to the Host Software Cluster.⌋

[SWS_SwCluC_02224]

Upstream requirements: SRS_SwCluC_00214

⌈The Os High Proxy shall require a service resource SwCluCOsProxyOsSpinlock, which is used in the Application Software Cluster, and needs to be matched to the Host Software Cluster.⌋

[SWS_SwCluC_02225]

Upstream requirements: SRS_SwCluC_00202, SRS_SwCluC_00214

⌈The Os High Proxy shall provide the following APIs for the Application Software Cluster:

- GetResource
- ReleaseResource
- SuspendOSInterrupts
- ResumeOSInterrupts
- GetSpinlock
- ReleaseSpinlock
- GetApplicationID
- GetCoreID
- CallTrustedFunction

Each of these APIs corresponds to a service resource in the Os Low Proxy. There will be an entry in the Binary Manifest, to require these service resources, which are described here: SwCluCOsProxy.⌋

[SWS_SwCluC_02279]

Upstream requirements: SRS_SwCluC_00214

⌈If the configuration parameter SwCluCOsProxyIdTranslationEnabled is enabled, the Os High Proxy shall support the translation of IDs (CoreId, OsApplicationId) configured in the Host Software Cluster into Application Software Cluster specific local IDs.⌋

Rationale: This is to avoid larger look-up tables required to fetch the correct ID configured in the Host Software Cluster for the use case where only a subset of the Host Software Cluster OS configuration is provided to the Application Software Clusters.

[SWS_SwCluC_02280]

Upstream requirements: SRS_SwCluC_00214

⌈If the configuration parameter `SwCluCOsProxyIdTranslationEnabled` is enabled, the ID reported by the following OS APIs for the `Application Software Clusters`:

- `GetApplicationID`
- `GetCoreID`

shall correspond to the `Application Software Cluster` specific local IDs. (See [SWS_SwCluC_02279])⌋

[SWS_SwCluC_CONSTR_02281] Consistent ID configuration between `Host Software Clusters` and `Application Software Clusters`

Upstream requirements: SRS_SwCluC_00214

⌈In case the parameter `SwCluCOsProxyIdTranslationEnabled` is set to `false` the configuration of core ID and application ID values in the according `Application Software Cluster` shall be identically to the ID values configured in the `Host Software Cluster`.⌋

[SWS_SwCluC_02226]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os High Proxy` shall NOT support the following APIs for the `Application Software Cluster`.

- `StartScheduleTable`
- `StopScheduleTable`
- `DisableAllInterrupts`
- `EnableAllInterrupts`
- `SuspendAllInterrupts`
- `ResumeAllInterrupts`

⌋

[SWS_SwCluC_02227]

Upstream requirements: SRS_SwCluC_00214

⌈The configuration of the `Os High Proxy` for a complete `Application Software Cluster` shall be specified with a set of ECUC-parameters defined here: `SwCluCOsProxy`.⌋

7.4.3.1.5 OS Low Proxy

[SWS_SwCluC_02250]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Low Proxy` shall only be provided by the `Host Software Cluster`.⌋

[SWS_SwCluC_02251]

Upstream requirements: SRS_SwCluC_00214

⌈The configuration of the `Os Low Proxy` for a complete `Host Software Cluster` shall be specified with a set of ECUC-parameters defined here: `SwCluCOsProxy`. The configuration of the `Os Low Proxy` shall be derived from the implemented configuration for the Os in the `Host Software Cluster`.⌋

[SWS_SwCluC_02252]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Low Proxy` shall provide a socket for the `Os High Proxys` of the `Application Software Clusters`. For each `Application Software Cluster`, a corresponding service resource `SwCluCOsProxyOsBaseSocket` and configuration shall be provided.⌋

[SWS_SwCluC_02253]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Low Proxy` shall provide for each `SwCluCOsProxyOsBaseSocket` a `SwCluC_OsProxy_Init` API, to initialize the `Os High Proxy` of an `Application Software Cluster`. This shall be one element of the service resource `SwCluCOsProxyOsBaseSocket`.⌋

For example, an Init-Callback could be provided for each `Application Software Cluster`, and called from the `Os Low Proxy` during initialization phase of the `Host Software Cluster`.

[SWS_SwCluC_02254]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Low Proxy` shall provide an API to start the `Os High Proxy` of an `Application Software Cluster`. This shall be one element of the service resource `SwCluCOsProxyOsBaseSocket`.⌋

For example, a Startup hook could be provided for each `Application Software Cluster`.

[SWS_SwCluC_02255]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall provide an API to shutdown the Os High Proxy of an Application Software Cluster. This shall be one element of the service resource SwCluCOsProxyOsBaseSocket.⌋

For example, a shutdown hook could be provided for each Application Software Cluster.

[SWS_SwCluC_02256]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall be able to provide an API to connect IOC from Os High Proxy of an Application Software Cluster. This shall be one element of the service resource SwCluCOsProxyOsBaseSocket.⌋

For example, an IOC-Init callback could be provided for each Application Software Cluster, and called from the Os Low Proxy, if a connection is established.

Dispatchers

Please note section 'Overview of Os Proxy' 7.11.

Example: The Host Software Cluster has a full configuration of the Os, and is prepared to host serveral Application Software Cluster, with their local RTE, to be able to execute the runnables of the Application Software Cluster's software components. This preconfiguration will preserve positions in an OS Task, like for events of runnables. Instead of directly executing a runnable that belongs to an Application Software Cluster, a so called Dispatcher Runnable is scheduled in the OsTask on the Host Software Cluster and will be executed. This Dispatcher acts as a link to Application Software Clusters task container, which will then execute all mapped runnables for this specific part of the task.

A connection between Software Clusters is established, when resource Id in the Binary Manifests matches. This is necessary for service resources of the SwCluCOsProxyOsTask.

[SWS_SwCluC_02259]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsTask, for each Application Software Clusters task configuration that the system shall be prepared for. For this resource, an entry in the Binary Manifest shall be created, to be able to link it to the Os High Proxy task in the Host Software Cluster.⌋

Please note that the SwCluCOsProxyOsTask container has a reference SwCluCOsProxyOsTaskRef to a OsTask container. This reference shall be used in the Host Software Cluster and the Application Software Cluster to obtain the OsTask configuration relevant for the guard value calculation. Hence, it is ensured that

Host Software Cluster and Application Software Cluster's OsTask configuration is compatible. See as well [SWS_SwCluC_90004].

[SWS_SwCluC_02284] SwCluCBManifProvideResourceEntryGroup for SwCluCOsProxyOsTask

Upstream requirements: SRS_SwCluC_00214

⌈For the Os Low Proxy, there shall be a SwCluCBManifProvideResourceEntryGroup, whose SwCluCBManifResourceTypeRef shall refer to a SwCluCBManifResourceType that contains one SwCluCBManifNotifierHandle, with SwCluCBManifNativeHandleType set to FUNCTION_REFERENCE.

The SwCluCBManifMultipleNotifierSupport shall be set to MULTIPLE_NOTIFIER_SETS.⌋

[SWS_SwCluC_02285] Semantics of SwCluCOsProxyMaxNumberOfCallee

Upstream requirements: SRS_SwCluC_00214

⌈For each SwCluCOsProxyOsTask, the Os Low Proxy shall put a SwCluCBManifProvideResourceEntry into the binary manifest.

If the proxy task's SwCluCOsProxyOsTaskDispatcher has SwCluCOsProxyMaxNumberOfCallee set to a value bigger than 1, the provide entry shall be placed in the SwCluCBManifProvideResourceEntryGroup defined in [SWS_SwCluC_02284] and the SwCluCBManifMaxNumberOfNotifierSets shall be set to the value of SwCluCOsProxyMaxNumberOfCallee.⌋

Note: If SwCluCOsProxyMaxNumberOfCallee is not set, or set to 1, it's not mandatory to place it into this SwCluCBManifProvideResourceEntryGroup. It's possible to have an additional SwCluCBManifProvideResourceEntryGroup, where SwCluCBManifMultipleNotifierSupport is set to SINGLE_NOTIFIER_SET. For MULTIPLE_NOTIFIER_SETS, there is always an additional element in the provide handle array (to store the number of handles in use), even if an SwCluCOsProxyOsTask does not use the feature. Having two SwCluCBManifProvideResourceEntryGroups allows to selectively enable the feature - SwCluCOsProxyOsTasks with SwCluCOsProxyMaxNumberOfCallee > 1 use the MULTIPLE_NOTIFIER_SETS SwCluCBManifProvideResourceEntryGroup, other SwCluCOsProxyOsTasks use the SINGLE_NOTIFIER_SET SwCluCBManifProvideResourceEntryGroup.

[SWS_SwCluC_02271]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall have a service resource SwCluCOsProxyFunctionDispatcher, for each function dispatcher provided in the Host Software Cluster. For this resource, an entry in the Binary Manifest shall be created, to be able to link the function dispatcher to its Os High Proxy implementation in the Application Software Cluster.⌋

Example: The Host Software Cluster has a full configuration of the BSW. To initialize all modules properly, a call to Application Software Cluster might be necessary. For example, to initialize a local RAM area there. Since this function or runnable might not be known by the RTE as runnable of a software component, this function can be called via the Os Proxy function dispatcher. This dispatcher can be configured in the Host Software Cluster, to be called from ECUM or an Ini-Task. Each dispatcher provides its own service resource in the Binary Manifest. If a successful link to another Software Cluster is established, the call can be executed from the Host Software Cluster into the Application Software Cluster.

In an older version of AUTOSAR, there was the possibility to have multiple Dispatchers for one SwCluCOsProxyOsTask. As a consequence, it was required to configure Dispatchers in both the Host Software Cluster and the Application Software Cluster. This was changed with R25-11. There is now only one Dispatcher per SwCluCOsProxyOsTask. SwCluCOsProxyOsTaskDispatcher describes the runnable in the Os Low Proxy that will call the OsTask (via the SwCluCOsProxyOsTask Binary Manifest Entry) in the Application Software Cluster. The Os High Proxy configuration does not contain an SwCluCOsProxyOsTaskDispatcher. The Dispatcher has no Binary Manifest entry and no longer needs a CpSoftwareClusterServiceResource.

Scheduling of dispatchers in Host Software Cluster

[SWS_SwCluC_02272]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall create one partition specific Service Software Component per configured EcucPartition for the dispatcher, with the name SwCluC_Dispatcher_<EcucPartition shortName>.⌋

[SWS_SwCluC_02273]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall create one RunnableEntity per configured SwCluCOsProxyOsTaskDispatcher container in the partition specific Service Software Component according [SWS_SwCluC_02272].

The EcucPartition is determined by the parameterSwCluCOsProxyEcucPartitionRef, if configured or alternatively by the EcucPartition to which the related OsTask belongs. The attributes of the RunnableEntity shall be set as following:

- minimumStartInterval is set to the parameter value SwCluCOsProxyMinimumStartInterval
- symbol is set to the name of the task dispatcher's C-function used in implementation.
- canBeInvokedConcurrently is set to false

⌋

[SWS_SwCluC_02278]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Low Proxy` shall create one `InitEvent` per configured `SwCluCOsProxyDispatcherInitEvent` container in the partition specific `Service Software Component` according [SWS_SwCluC_02272]. The attributes of the `InitEvent` shall be set as following:

- `shortName` is set according the container's `shortName`
- `startOnEvent` is set to the `RunnableEntity` according [SWS_SwCluC_02273] related to the owning `SwCluCOsProxyOsTaskDispatcher` container.

⌋

[SWS_SwCluC_02274]

Upstream requirements: SRS_SwCluC_00214

⌈The `Os Low Proxy` shall create one `BackgroundEvent` per configured `SwCluCOsProxyDispatcherBackgroundEvent` container in the partition specific `Service Software Component` according [SWS_SwCluC_02272]. The attributes of the `BackgroundEvent` shall be set as following:

- `shortName` is set according the container's `shortName`
- `startOnEvent` is set to the `RunnableEntity` according [SWS_SwCluC_02273] related to the owning `SwCluCOsProxyOsTaskDispatcher` container.
- `disabledMode`(s) are created according the `SwCluCOsProxyDisabledModeInstanceRef`(s) where for each different `PPortPrototype` an `RPortPrototype` is created at the owing `Service Software Component` according [SWS_SwCluC_02272] and used as `contextPort`.

⌋

[SWS_SwCluC_02275]

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00478

⌈The `Os Low Proxy` shall create one `TimingEvent` per configured `SwCluCOsProxyDispatcherTimingEvent` container in the partition specific `Service Software Component` according [SWS_SwCluC_02272]. The attributes of the `TimingEvent` shall be set as following:

- `shortName` is set according the container's `shortName`
- `startOnEvent` is set to the `RunnableEntity` according [SWS_SwCluC_02273] related to the owning `SwCluCOsProxyOsTaskDispatcher` container
- `period` value is set according the `SwCluCOsProxyTimingEventPeriod` parameter

- disabledMode(s) are created according the SwCluCOsProxyDisabledModeInstanceRef(s) where for each different PPortPrototype an RPortPrototype is created at the owing Service Software Component according [SWS_SwCluC_02272] and used as contextPort.

⌋

[SWS_SwCluC_02276]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall create one SwcModeSwitchEvent per configured SwCluCOsProxyDispatcherModeSwitchEvent container in the partition specific Service Software Component according [SWS_SwCluC_02272]. The attributes of the SwcModeSwitchEvent shall be set as following:

- shortName is set according the container's shortName
- startOnEvent is set to the RunnableEntity according [SWS_SwCluC_02273] related to the owning SwCluCOsProxyOsTaskDispatcher container
- activation value is set according the SwCluCOsProxyDispatcherActivation parameter
- modes are created according the SwCluCOsProxyModeInstanceRef(s) where for each different PPortPrototype an RPortPrototype is created at the owing Service Software Component according [SWS_SwCluC_02272] and used as contextPort.
- disabledMode(s) are created according the SwCluCOsProxyDisabledModeInstanceRef(s) where for each different PPortPrototype an RPortPrototype is created at the owing Service Software Component according [SWS_SwCluC_02272] and used as contextPort.

⌋

[SWS_SwCluC_02277]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall create one ExternalTriggerOccurredEvent per configured SwCluCOsProxyDispatcherExternalTriggerOccurredEvent container in the partition specific Service Software Component according [SWS_SwCluC_02272]. The attributes of the ExternalTriggerOccurredEvent shall be set as following:

- shortName is set according the container's shortName
- startOnEvent is set to the RunnableEntity according [SWS_SwCluC_02273] related to the owning SwCluCOsProxyOsTaskDispatcher container
- trigger is set according the SwCluCOsProxyTriggerInstanceRef where for each different PPortPrototype an RPortPrototype is created at the

owing Service Software Component according [SWS_SwCluC_02272] and used as contextRPort.

- disabledMode(s) are created according the SwCluCOsProxyDisabledModeInstanceRef(s) where for each different PPortPrototype an RPortPrototype is created at the owing Service Software Component according [SWS_SwCluC_02272] and used as contextPort.

⌋

OsApplication

[SWS_SwCluC_02262]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsApplication, for each OsApplication that is configured in the Host Software Cluster, and prepared for use in Application Software Clusters.⌋

OsResource

[SWS_SwCluC_02263]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsResource, for each OsResource used in Application Software Clusters.⌋

OsSpinlock

[SWS_SwCluC_02264]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsSpinlock, for each OsSpinlock used in Application Software Clusters.⌋

OS APIs

[SWS_SwCluC_02265]

Upstream requirements: SRS_SwCluC_00214

⌈The Os Low Proxy shall provide a basic service implementation for the following APIs, to be used and connected with an Application Software Cluster:

- ActivateTask
- GetResource
- ReleaseResource
- SuspendOSInterrupts
- ResumeOSInterrupts

- ReleaseResource
- GetSpinlock
- ReleaseSpinlock
- GetApplicationID
- Schedule
- CallTrustedFunction
- GetCoreID

These APIs shall be provided with a corresponding service resource from SwCluCOsProxy.⌋

The APIs ChainTask and TerminateTask are fully implemented in the Os High Proxy. Hence the Os Low Proxy does not provide any specific support for those functions.

Unconnected Service Resources

[SWS_SwCluC_02266]

Upstream requirements: SRS_SwCluC_00213

⌈The Os Low Proxy shall provide default handles for resources, in case a connection could not be established successfully between Application Software Cluster and the Host Software Cluster.⌋

For example, if an Application Software Cluster is missing, the task-dispatcher in the host calls an empty function.

Please note: One property of a clustered system is that parts shall be independently buildable. Another property is that some parts can be absent. Therefore, "default stubs" are needed for non-local APIs. But in case of missing interfaces to Os, Application Software Clusters cannot run successfully. The usage of a suitable mechanism to detect this case is advised. To achieve a robust system, the Host Software Cluster is still running in such a state.

7.4.3.2 NvM Proxy

The specified NvM Proxy has the following underlying design principle.

In the Application Software Cluster, NV Blocks supported by the NvM High Proxy are configured with EcucModuleConfigurationValues for a NvM EcucModuleDef, with according NvMBlockDescriptors containers. In the Application Software Cluster and in the Host Software Cluster, the configuration of SwCluCNvMProxyNvBlock containers map the NvMBlockDescriptors to CpSoftwareClusterServiceResources, defining each the globalResourceId

and the isMandatory attribute, relevant for the corresponding Resource Entry in the Binary Manifest.

The connection between the NvM High Proxy and the NvM Low Proxy utilizes the explicit synchronization, as specified in document [6]. The explicit synchronization has the advantage that the addresses of RAM Blocks and ROM blocks of Application Software Cluster do not have to be known by the NVRAM Manager in the Host Software Cluster. In addition, the function interface can be used to introduce additional functionality, like the NV Block length adjustment ([SWS_SwCluC_02105]). On the other hand, this design principle cannot support temporary RAM blocks, since the NvM configuration in the Host Software Cluster may hold a different size than the NvM configuration in the Application Software Cluster.

[SWS_SwCluC_CONSTR_02134]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈The NvM High Proxy shall reject configurations of temporary RAM blocks.⌋

Note: this means only NVRAM blocks configured with a permanent RAM block or explicit synchronization callbacks are supported in the NvM High Proxy.

In addition, the Host Software Cluster should not have a functional dependency to the content stored in NV Blocks of the Application Software Cluster. Furthermore, the NvM High Proxy shall be able to initialize the RAM Block, even if the related NV Block is not connected to a Host Software Cluster. This requires the existence of ROM block default values.

[SWS_SwCluC_CONSTR_02135]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall reject configurations, where no ROM block is configured via the parameter NvMRomBlockDataAddress, or the parameter NvMBlockDescriptor.NvMInitBlockCallback.⌋

[SWS_SwCluC_CONSTR_02141]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall reject configurations, where the NV Block length given by NvMNvBlockLength is larger than the maximum size foreseen in the Host Software Cluster, given by SwCluCNvMProxyNvBlockMaxLength.⌋

Please note that SwCluCNvMProxyNvBlockMaxLength is also part of the guard value, and therefore protected from unilateral changes.

7.4.3.2.1 Enable NvM Proxy Generation

[SWS_SwCluC_02101]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00204

⌈The NvM High Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationNvM is set to HIGH_PROXY.⌋

[SWS_SwCluC_02144]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00204

⌈The NvM Low Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationNvM is set to LOW_PROXY.⌋

7.4.3.2.2 General NvM Proxy functionality

[SWS_SwCluC_02104]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Proxy shall support that the BlockIds of the NvM of Host Software Cluster can change, without reconfiguration or rebuild of the Application Software Cluster using these NV blocks.⌋

[SWS_SwCluC_02105]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Proxy shall support that the NV block length used in the Application Software Cluster can be <= the NV block length configured in the NvM of the Host Software Cluster. In case NV block length in Application Software Cluster is smaller, the NV block is filled with default values. The default fill value of a single byte is defined by the configuration parameter SwCluCNvMProxyNvBlockFillValue.⌋

[SWS_SwCluC_02106]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Proxy shall support that the NV block length used in the Application Software Cluster can be changed (in the range 0 <= NV block length configured in NvM), without reconfiguration or rebuild of the Host Software Cluster.⌋

[SWS_SwCluC_02112]

Upstream requirements: SRS_SwCluC_00206

⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_SingleBlockCallbackFunction or JobFinished operation, when the according instance of the NvM Low Proxy is called.⌋

[SWS_SwCluC_02122]

Upstream requirements: SRS_SwCluC_00206

⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_InitBlockCallbackFunction or InitBlock operation, when the according instance of the NvM Low Proxy is called.⌋

[SWS_SwCluC_02123]

Upstream requirements: SRS_SwCluC_00206

⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_ReadRamBlockFromNvm function or ReadRamBlockFromNvM operation, when the according instance of the NvM Low Proxy is called.⌋

[SWS_SwCluC_02124]

Upstream requirements: SRS_SwCluC_00206

⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_WriteRamBlockToNvm function or WriteRamBlockToNvM operation, when the according instance of the NvM Low Proxy is called.⌋

7.4.3.2.3 Configuration ID check

In case of a non-clustered AUTOSAR ECU the configuration ID check is centrally implemented in NVRAM Manager when it is enabled via NvMDynamicConfiguration is set to TRUE.

If a configuration ID mismatch is detected, for NVRAM blocks which are configured with NvMResistantToChangedSw == FALSE, default data are loaded independent of the validity of an assigned RAM or NV block. This feature ensures an initialization of NV blocks in case of a changed NVRAM memory layout, i.e., if a block is added or removed, or if its size or type is changed.

In case of an AUTOSAR ECU using independently reprogrammable Software Clusters, a common configuration ID has the drawback of unintended reprogramming dependencies. In case of a NVRAM memory layout change caused by one Software Cluster, the non-resistant-to-changed-SW NV blocks of all other Software Clusters needs to be set to default as well. In addition, such a configuration ID change can only be initiated via an update of the Host Software Cluster.

To enable a check on changed NVRAM memory layout per Application Software Cluster the NvM High Proxy supports individual configuration IDs per Application Software Cluster.

Since the NvM High Proxy cannot directly interfere with the NvM_ReadAll execution it has to implement its configuration ID checking in the execution of the NvM_ReadRamBlockFromNvm triggered by the execution of the multi block request triggered by

NvM_ReadAll. This configuration ID check is only active after the initialization of the NvM Proxy till the NvM_ReadAll multi block request is successfully finished.

The NvM Proxy gets informed about the successful finish of NvM_ReadAll by the call of SwCluC_NvMProxy_MultiBlockReadAllJobFinished from BswM via an appropriately configured action list.

The time span between NvM Proxy initialization and the call of SwCluC_NvMProxy_MultiBlockReadAllJobFinished is called NvM startup phase.

To ensure a working behavior of the NvM Proxy in the SwCluC, the NVRAM Manager of the Host Software Cluster has to treat all NvBlocks of an Application Software Cluster as NvMResistantToChangedSw == TRUE. This behavior can be ensured by the appropriate configuration of the NVRAM Manager.

[SWS_SwCluC_CONSTR_02145] Application Software Cluster's NvBlocks are treated as 'resistant to changed SW' by NvM in Host Software Cluster

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00167

⌈For any NVRAM Block belonging to an Application Software Cluster (NvM Low Proxy of the SwCluC occurs as block user) the parameter NvMResistantToChangedSw shall be set to TRUE in the Host Software Cluster.⌋

Note: This includes the NVRAM Blocks used to store the configuration ID of a specific Application Software Cluster

In addition, the NvBlocks in the Host Software Cluster have to be configured such, that the Configuration ID block of an Application Software Cluster is restored before the regular NvBlocks of that Application Software Cluster. Since the NvM ensures the processing order of NvM Blocks during a NvM_ReadAll job by the Block Id ([SWS_NvM_00244]). This enables that the NvM High Proxy can process its Config Block ID handling before that recovery of the regular NvBlocks.

[SWS_SwCluC_CONSTR_02146] Block IDs of Application Software Cluster's Configuration ID blocks in Host Software Cluster

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00167

⌈In the NvM module of the Host Software Cluster the Configuration ID block of an Application Software Cluster shall be configured with a lower block ID than all the other NvBlocks of the identical Application Software Cluster.⌋

Please note, that the configured Block IDs of an Application Software Cluster's NvBlock can differ in the configuration in the NvM High Proxy and the NvM of the Host Software Cluster.

The Block ID numbers are translated during the API calls from Application Software Cluster to the Host Software Cluster.

Hence, a valid configuration of Block IDs in the Host Software Cluster can look like following example as shown in figure 7.12.

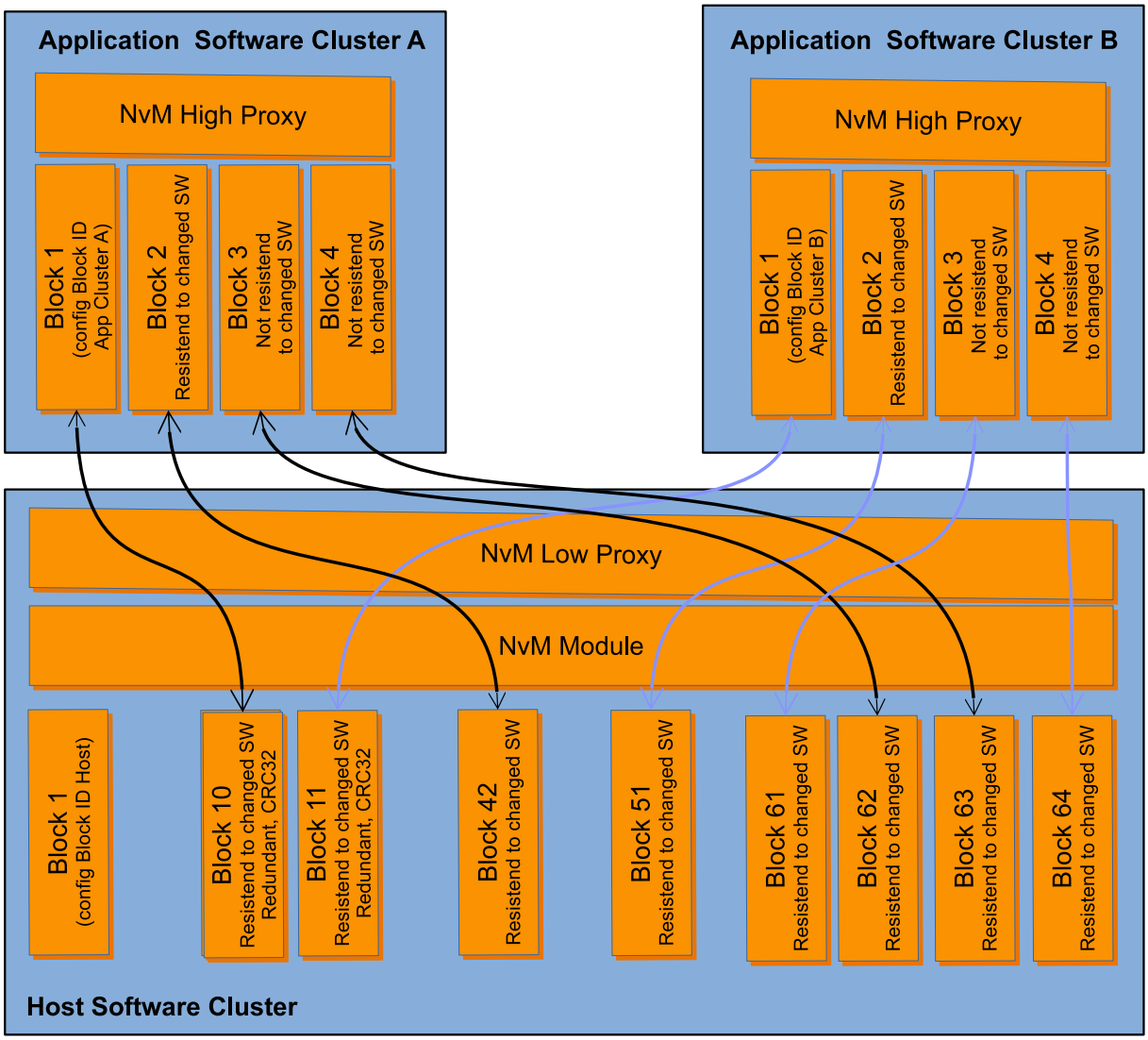

Figure 7.12: Example configuration of NvBlocks used by Application Software Clusters

[SWS_SwCluC_02147]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy of a specific Application Software Cluster shall enable the configuration ID check, if the parameter NvMDynamic-Configuration is set to TRUE in this Application Software Cluster.⌋

Note: The parameter NvMDynamicConfiguration is set in the NvM module configuration of the respective Application Software Cluster.

Like in the configuration of the NvM module in the Host Software Cluster, the block ID 1 is reserved for the Configuration ID block in each Application Software Cluster. This Configuration ID block to contain the stored NVRAM configuration ID.

[SWS_SwCluC_CONSTR_02148] `Configuration ID block` uses always block ID 1

Upstream requirements: SRS_SwCluC_00206

⌈In the configuration of the `NvM High Proxy`, the NVRAM block with the block ID 1 shall be reserved for the `Configuration ID block` to contain the stored `NVRAM configuration ID`.⌋

[SWS_SwCluC_02149]

Upstream requirements: SRS_SwCluC_00206

⌈The `NvM High Proxy` of a specific `Application Software Cluster` shall implement the `Configuration ID block` as redundant type with `CRC32`.⌋

[SWS_SwCluC_CONSTR_02160] NvBlock configuration of `Application Software Cluster`'s `Configuration ID blocks` in `Host Software Cluster`

Upstream requirements: SRS_SwCluC_00206

⌈In the NvM module of the `Host Software Cluster` the `Configuration ID block` of an `Application Software Cluster` shall be configured with `NvMBlockManagementType` == `NVM_BLOCK_REDUNDANT` and `NvMBlockCrcType` == `NVM_CRC32`.⌋

[SWS_SwCluC_02150]

Upstream requirements: SRS_SwCluC_00206

⌈The job of the function `NvM_WriteBlock` for the `Configuration ID block` shall write the compiled `NVRAM configuration ID` to the stored `NVRAM configuration ID` (in the `Configuration ID blocks`) of this `Application Software Cluster`.⌋

[SWS_SwCluC_02151]

Upstream requirements: SRS_SwCluC_00206

⌈The job of the function `NvM_WritePRAMBlock` for the `Configuration ID block` shall write the compiled `NVRAM configuration ID` to the stored `NVRAM configuration ID` (in the `Configuration ID blocks`) of this `Application Software Cluster`.⌋

[SWS_SwCluC_02152]

Upstream requirements: SRS_SwCluC_00206

⌈The `NvM_ReadRamBlockFromNvm` call for the `Configuration ID block` shall validate the configuration ID by comparing the stored `NVRAM configuration ID` of this `Application Software Cluster` vs. the compiled `NVRAM configuration ID` of this `Application Software Cluster`.⌋

In case the NvM module cannot read the Configuration ID block of an Application Software Cluster because the corresponding NV block is empty or invalidated the NvM Module will call the respective NvM_InitBlockCallbackFunction for the Configuration ID block.

[SWS_SwCluC_02153]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM_ReadRamBlockFromNvm call for the Configuration ID block shall treat the result of the configuration ID check as a configuration ID mismatch.⌋

Note: The later update and validation of the Configuration ID block is handled in [SWS_SwCluC_02157].

[SWS_SwCluC_02154]

Upstream requirements: SRS_SwCluC_00206

⌈The job of the function NvM_GetErrorStatus reports for the Configuration ID block NVM_REQ_NOT_OK in case of configuration ID mismatch.⌋

Note: this ensures a compatible behavior as [SWS_NvM_00306]

[SWS_SwCluC_02155]

Upstream requirements: SRS_SwCluC_00206

⌈The job of the function NvM_GetErrorStatus reports for the Configuration ID block NVM_REQ_NV_INVALIDATED till the Configuration ID block of this Application Software Cluster was restored valid from NvRam⌋

Note: this ensures a compatible behavior as [SWS_NvM_00672]

[SWS_SwCluC_02156]

Upstream requirements: SRS_SwCluC_00206

⌈The function NvM_ReadRamBlockFromNvm shall get the ROM Block default values and copy them to application if

- configuration ID check is enabled (NvMDynamicConfiguration is set to TRUE)

  AND

- a configuration ID mismatch was detected

  AND

- for the related NvBlock in the Application Software Cluster the NvMResistantToChangedSw is set to FALSE

  AND

- NvM Proxy is in NvM startup phase

In all other cases the function NvM_ReadRamBlockFromNvm shall process the normal data copy functionality from NvM to application⌋

[SWS_SwCluC_02157]

Upstream requirements: SRS_SwCluC_00206

⌈The function SwCluC_NvMProxy_MultiBlockReadAllJobFinished shall update the Configuration ID block according to the new (compiled) configuration ID and validate the according NVRAM Block if a configuration ID mismatch occurred and if the NvMDynamicConfiguration == TRUE.⌋

Note: The validation NVRAM block with config ID triggers a CRC recalculation and enables that it gets written during next NvM_WriteAll.

7.4.3.2.4 NvM High Proxy

[SWS_SwCluC_02102]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall provide the header file NvM.h.⌋

[SWS_SwCluC_02103]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall provide the symbolic name values for NvBlocks, for NvM users in the Application Software Cluster.⌋

[SWS_SwCluC_02125]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00202

⌈The NvM High Proxy shall provide the functions for single block requests

- NvM_SetDataIndex
- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_GetErrorStatus
- NvM_SetRamBlockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock

- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

for NvM users in the Application Software Cluster.⌋

[SWS_SwCluC_02136]

Upstream requirements: SRS_SwCluC_00206

⌈The functions

- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults

of the NvM High Proxy shall only accept single block requests, where the pointer to the RAM data block is set to NULL_PTR. Otherwise, E_NOT_OK is returned, and the request is discarded.⌋

Callback Functions

[SWS_SwCluC_02109]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_SingleBlockCallbackFunction, according to [ECUC_NvM_00562], [ECUC_NvM_00506] and [ECUC_NvM_00559].⌋

[SWS_SwCluC_02110]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_PNJF_{Block} port for NvMNotifyJobFinished notification.⌋

[SWS_SwCluC_02115]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_InitBlockCallbackFunction, according to [ECUC_NvM_00561], [ECUC_NvM_00116] and [ECUC_NvM_00559].⌋

[SWS_SwCluC_02116]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_PNIB_{Block} port for NvMNotifyInitBlock notification.⌋

[SWS_SwCluC_02117]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_ReadRamBlockFromNvm, according to [ECUC_NvM_00521].⌋

[SWS_SwCluC_02118]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_WriteRamBlockToNvm, according to [ECUC_NvM_00520].⌋

[SWS_SwCluC_02119]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall support the optional configuration of a NvM_PM_{Block} port for NvMMirror notification.⌋

Multiple Partitions

[SWS_SwCluC_02120]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM High Proxy shall create one partition specific Service Software Component per configured EcucPartition, with the name NvM_<EcucPartition shortName>.⌋

[SWS_SwCluC_02121]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00201

⌈The NvM High Proxy shall provide the Ports belonging to a specific NV Block

- NvM_PS_{Block}
- NvM_PAdmin_{Block}
- NvM_PNIB_{Block}
- NvM_PNJF_{Block}
- NvM_PM_{Block}

at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

Handling of unconnected blocks

[SWS_SwCluC_02107]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈In case a NV block configured in the NvM High Proxy is not connected to any NvM Low Proxy, the NvM High Proxy shall initialize the Nv RAM Block, with the default values given by NvMRomBlockDataAddress and / or NvMInitBlockCallback.⌋

[SWS_SwCluC_02113]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈In case a NV block configured in the NvM High Proxy is not connected to any NvM Low Proxy, the function NvM_GetErrorStatus of the NvM High Proxy shall still return E_OK, and the request result is set to NVM_REQ_RESTORED_FROM_ROM.⌋

[SWS_SwCluC_02114]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈In case a NV block configured in the NvM High Proxy is not connected to any NvM Low Proxy, the functions

- NvM_SetDataIndex
- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_SetRamBlockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock
- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

of the NvM High Proxy shall return E_NOT_OK.⌋

7.4.3.2.5 NvM Low Proxy

As the NvM BSW module already ensures that callbacks are partition specific (i.e. the callback function is called from the correct partition, see [SWS_NvM_00969]), the NvM Low Proxy is not required to implement features to switch partitions as described in chapter 7.4.2.2.

[SWS_SwCluC_02132]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Low Proxy shall provide the entry functions for single block requests, according to [SWS_SwCluC_02125],⌋

[SWS_SwCluC_02111]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Low Proxy shall provide an instance of a NvM_SingleBlockCallback-Function per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋

[SWS_SwCluC_02127]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Low Proxy shall provide an instance of a NvM_InitBlockCallback-Function per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋

[SWS_SwCluC_02128]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Low Proxy shall provide an instance of a NvM_ReadRamBlockFromNvm per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋

[SWS_SwCluC_02129]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Low Proxy shall provide an instance of a NvM_WriteRamBlockToNvm per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋

[SWS_SwCluC_02133]

Upstream requirements: SRS_SwCluC_00206

⌈The callback instances according to [SWS_SwCluC_02111], [SWS_SwCluC_02127], [SWS_SwCluC_02128], and [SWS_SwCluC_02129], are provided at the EcucPartition referenced by SwCluCNvMProxyNvBlock.SwCluCNvMProxyNvBlock-EcucPartitionRef.

⌋

[SWS_SwCluC_CONSTR_02165] Consistent partion configuration for NvBlocks and NvM Block Proxies

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈In the Host Software Cluster the EcucPartition of a SwCluCNvMProxy-NvBlock (referenced via SwCluCNvMProxyNvBlock.SwCluCNvMProxyNvBlock-EcucPartitionRef) shall be identical to the EcucPartition of the related NvMBlockDescriptor (referenced via NvMBlockDescriptor.NvMBlockEcuc-PartitionRef) The related NvMBlockDescriptor is defined via SwCluCNv-MProxyNvBlock.SwCluCNvMProxyNvMBlockDescriptorRef.⌋

Handling of unconnected blocks

In case a NV Block configured in the NvM Low Proxy being unconnected, the NvM can still invoke the callback functions without any effect. But the NvM Low Proxy will not invoke any of the NvM APIs with the according Block ID autonomously.

[SWS_SwCluC_02161] Unconnected-behavior of NvM_SingleBlockCallbackFunction

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈The NvM_SingleBlockCallbackFunction of NvM Low Proxy shall return E_NOT_OK, if the corresponding NV Block is not connected to an instance in the NvM High Proxy.⌋

[SWS_SwCluC_02162] Unconnected-behavior of NvM_InitBlockCallbackFunction

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈The NvM_InitBlockCallbackFunction of NvM Low Proxy shall return E_NOT_OK, if the corresponding NV Block is not connected to an instance in the NvM High Proxy.⌋

[SWS_SwCluC_02163] Unconnected-behavior of NvM_ReadRamBlockFromNvm

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈The NvM_ReadRamBlockFromNvm of NvM Low Proxy shall return E_NOT_OK, if the corresponding NV Block is not connected to an instance in the NvM High Proxy.⌋

[SWS_SwCluC_02164] Unconnected-behavior of NvM_WriteRamBlockToNvm

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈The NvM_WriteRamBlockToNvm of NvM Low Proxy shall return E_NOT_OK, if the corresponding NV Block is not connected to an instance in the NvM High Proxy.⌋

[SWS_SwCluC_02108]

Upstream requirements: SRS_SwCluC_00206, SRS_SwCluC_00213

⌈In case a NV block configured in the NvM Low Proxy is not connected to any NvM High Proxy, the NvM Low Proxy implementation shall not modify the content of that NV Block.⌋

Note: An access by Dcm might still change the NV Block content in this case.

[SWS_SwCluC_02142]

Upstream requirements: SRS_SwCluC_00206

⌈The NvM Low Proxy implementation shall describe the NV blocks provided to NvM High Proxies via NvBlockNeeds, derived from the CpSoftwareClusterService-Resource.resourceNeeds.⌋

Note: [SWS_SwCluC_02142] ensures an automated configuration of the NVRAM Manager.

[SWS_SwCluC_02143]

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00101

⌈The NvM Low Proxy shall provide, for each SwCluCNvMBaseSocket, a SwCluC_NvMProxy_Init API, to initialize the NvM High Proxy of an Application Software Cluster. This shall be one element of the service resource SwCluCNvMBaseSocket.⌋

7.4.3.2.6 Error detection

[SWS_SwCluC_02137]

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If development error detection is enabled for NvM High Proxy, the APIs

- NvM_SetDataIndex
- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_GetErrorStatus
- NvM_SetRamBlockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock
- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

shall report the DET error NVM_E_UNINIT, when NVM High Proxy is not yet initialized.⌋

[SWS_SwCluC_02138]

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If development error detection is enabled for NvM High Proxy, the APIs

- NvM_SetDataIndex

- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_GetErrorStatus
- NvM_SetRamBlockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock
- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

shall report the DET error NVM_E_PARAM_BLOCK_ID, when the passed BlockId is out of range of the BlockIds configured in the NvM High Proxy.⌋

[SWS_SwCluC_02139]

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If development error detection is enabled for NvM High Proxy, the APIs

- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults

shall report the DET error NVM_E_PARAM_ADDRESS, when a RAM block address different than NULL_PTR is passed.⌋

7.4.3.3 Com Proxy

The specified Com Proxy has the following underlying design principle.

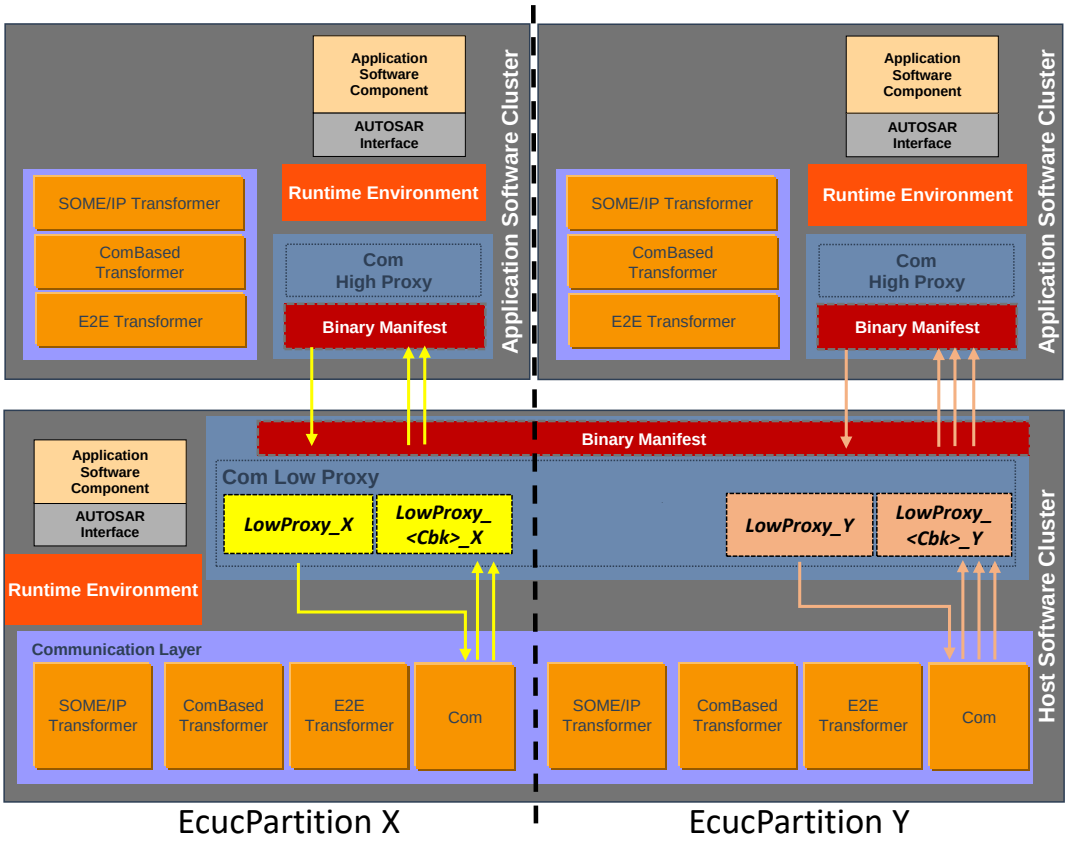

Figure 7.13: Overview of Com Proxy

The Application Software Cluster provides proxies for those parts of the Com implementation that are required to complete its functionality. This is called Com High Proxy.

The Host Software Cluster contains the real implementation of the Com functionality and makes it available to the Com High Proxy via the respective Com Low Proxy. Both Com High Proxy and Com Low Proxy are linked together based on the information available in Binary Manifest, which enables the execution of Com (Callback) functions from Application Software Cluster to Host Software Cluster and vice-versa.

The Com module can now have an arbitrary of users with the Com Low Proxy being one of those. It therefore relies on the usage of URI References (See [16] Chapter 2.3.6.6 *"Uri Reference"*) to be linked with its respective users in the model.

The Com Low Proxy as an ComUserModule is represented by one or several ComUserModuleCnf. To guarantee the compatibility between configurations of the Com module and the Com Proxy, required parameters and containers are defined by the ComUserUriDefSet. A ComUserModuleCnf is required for each EcucPartition, in which the Com Proxy requires direct access to Com APIs and provides notification

callbacks for transmission / reception. Those APIs depend on the configured data path in Com, either ComSignals / ComSignalGroups based (see Chapter 7.4.3.3.2) or ComIPdu based (see Chapter 7.4.3.3.3).

[SWS_SwCluC_02604]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support the invocation of Com APIs from an Application Software Cluster.⌋

[SWS_SwCluC_02605]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support the invocation of Com callback APIs from the Host Software Cluster.⌋

7.4.3.3.1 General Com Proxy functionality

7.4.3.3.1.1 Enable Com Proxy Generation

[SWS_SwCluC_02600]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00204

⌈The Com High Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationCom is set to HIGH_PROXY.⌋

[SWS_SwCluC_02601]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00204

⌈The Com Low Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationCom is set to LOW_PROXY.⌋

7.4.3.3.1.2 Com Proxy configuration principles

Com User represents a using entity of the Com module and configures the upper layer of Com. In particular the ComUserModuleCnf container provides the configuration relevant for the Com module's APIs as configured callbacks and callback handle Ids for ComSignals, ComSignalGroups and ComIPdus. It's possible that a single BSW module holds multiple Com Users to support a partition wise usage of Com stack in case of Com stack distribution to multiple cores. In the Host Software Clusters both RTE and Com Proxy of SwCluC act as "Com using module" and requiring such an upper layer configuration. Hence, Com User instances are provided by RTE and SwCluC's Com Proxy.

For RTE and SwCluC it's possible that either

- ECUC Partition specific callbacks

  or

- ECUC Partition agnostic callbacks

are implemented. Please not as well the according explantions in the documents

[7] Chapter 10.3.11.1 *"ECUC Partition Specific Com User"*,

[7] Chapter 10.3.11.2 *"ECUC Partition Unspecific Com User"*,

and [4] Chapter 8.16 *"Configuration of Com User"*.

In the first case for SwCluC's Com Low Proxy a SwCluCComProxyBaseSocket with a ComUserModuleCnf exists for each partition on which a Com Proxy in an Application Software Cluster exists.

In the second case for SwCluC's Com Low Proxy a SwCluCComProxyBaseSocket with a ComUserModuleCnf exists for each Application Software Cluster with a Com High Proxy.

Figure 7.14 illustrates the structure of the ECU configuration in Host Software Clusters and Application Software Clusters.

In the Host Software Cluster the Com module references all configured ComUserModuleCnf container of RTE and SwCluC module.

In the Application Software Cluster, the SwCluC's Com High Proxy substitutes the Com module and therefore the SwCluCComProxyBaseSocket references the ComUserModuleCnf of the RTE configuration.

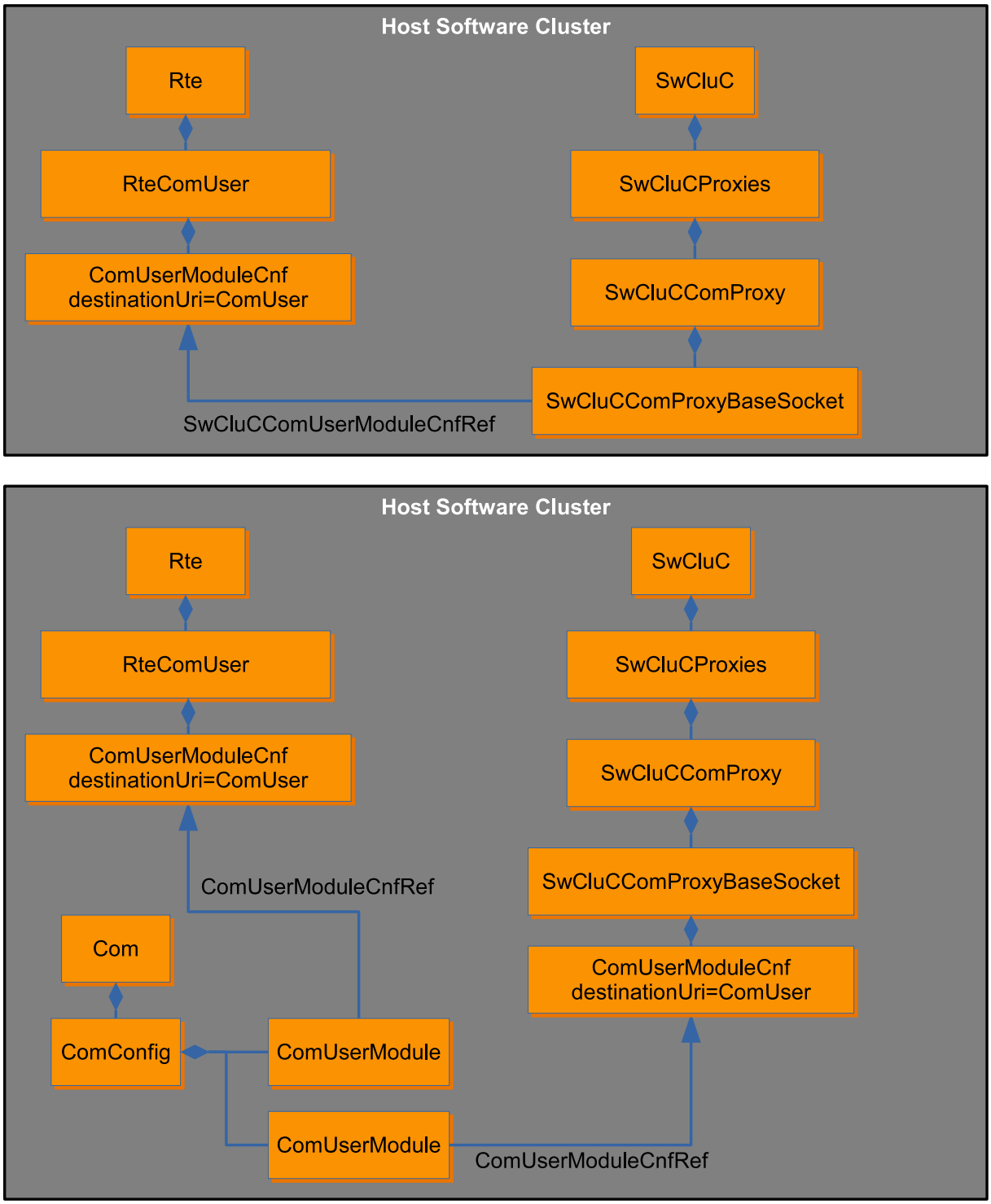

Figure 7.14: Overview of Com Proxy ECU configuration structure

Based on the scenario shown in figure 7.13 following configuration scenario shown in figure 7.15 would apply. In the `Host Software Cluster` both `RTE` and `SwCluC` have configured `Com Users` via `ComUserModuleCnf` containers. To support the two `Application Software Clusters` running on different partitions, the `Com Low`

Proxy requires two Com Users and consequently two ComUserModuleCnf containers. Consequently, for the Com module three ComUserModule containers are configured and each references the related ComUserModuleCnf container of RTE and SwCluC.

In each Application Software Cluster the RTE configures Com Users which are referenced by the according Com High Proxy configuration.

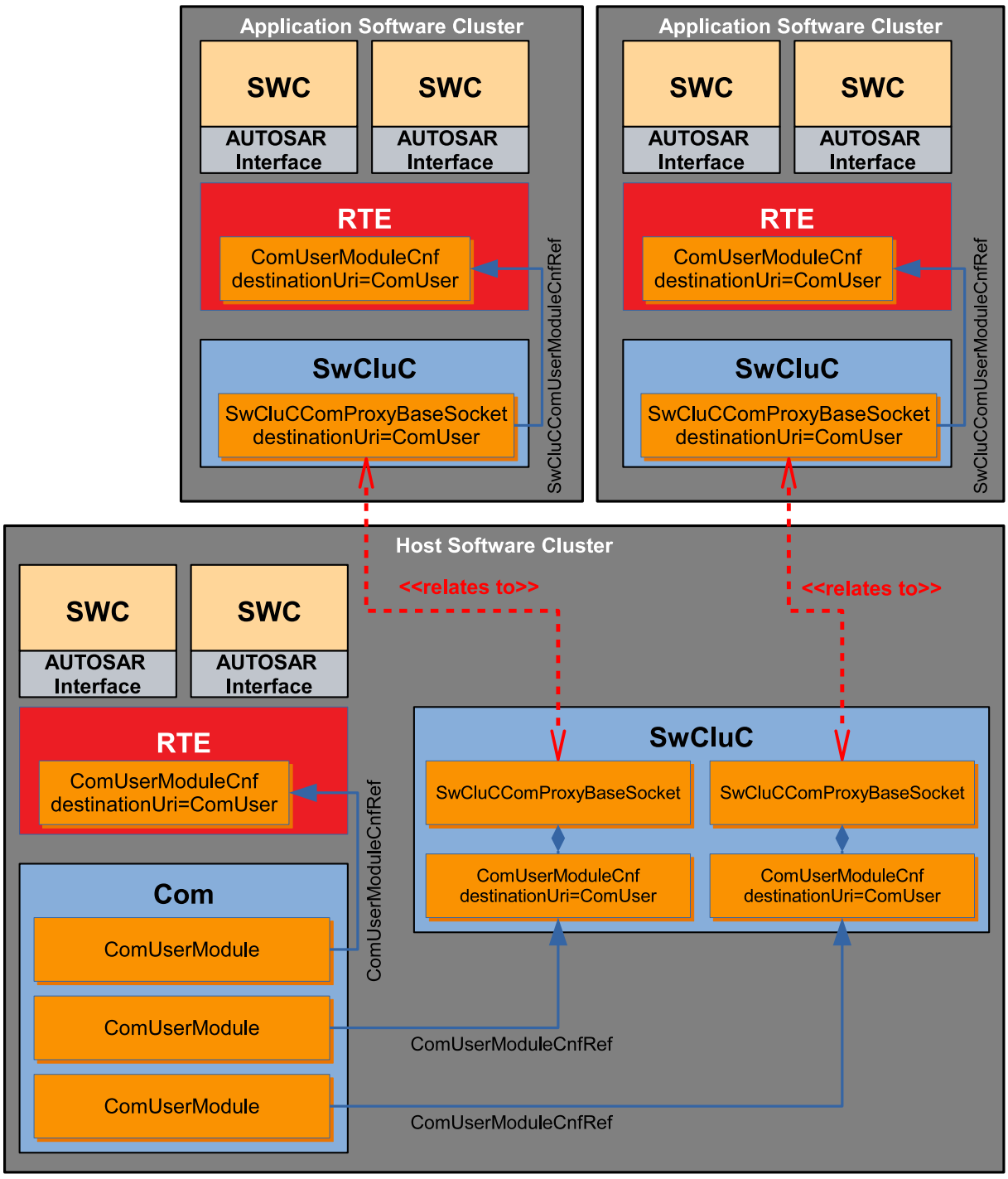

Figure 7.15: Com Proxy ECU configuration example

7.4.3.3.1.3 Com High Proxy

[SWS_SwCluC_02606]

Upstream requirements: SRS_SwCluC_00211

⌈The Com High Proxy shall provide the header file Com.h.⌋

7.4.3.3.1.4 Com Low Proxy

[SWS_SwCluC_02624]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide the header file SwCluC_ComProxy_Cbk.h.⌋

[SWS_SwCluC_CONSTR_02654]

Upstream requirements: SRS_SwCluC_00211

⌈The ComUserHeaderInclude parameter shall be configured as SwCluC_ComProxy_Cbk.h.⌋

The implementation of the Com Low Proxy may provide some basic infrastructure for the Com High Proxy in order to be operational. Such generic infrastructure services (including the access to Com APIs or the callbacks) can be linked between Application Software Cluster and Host Software Clusters by means of a Com base socket.

[SWS_SwCluC_02626]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide for each SwCluCComProxyBaseSocket a SwCluC_ComProxy_Init API, to initialize the Com High Proxy of an Application Software Cluster.⌋

[SWS_SwCluC_CONSTR_02667] Ensure that Com notification callbacks are available in the Application Software Cluster

Upstream requirements: SRS_SwCluC_00211

⌈The isMandatory attribute of the CpSoftwareClusterServiceResource referenced by an SwCluCComProxyBaseSocket instance shall be set to true.⌋

The rationale of the constraint [SWS_SwCluC_CONSTR_02667] is to ensure Com notification callbacks which are available in the Application Software Cluster will be invoked by the Host Software Cluster to ensure correct functionality w.r.t ComSignal and ComIPdu reception in that Application Software Cluster.

7.4.3.3.2 ComSignals / ComSignalGroups based proxy handling

If using a data path in Com that refers to ComSignals and/or ComSignalGroups, then a ComUserModuleCnf is required for each EcucPartition, in which the Com Proxy

- requires direct access to the Com APIs initiating transmission/reception requests of ComSignals and/or ComSignalGroups
- provides transmission/reception notification callbacks of ComSignals and/or ComSignalGroups

Effectively, a ComUserModuleCnf links a fixed set of notification callbacks in the Com to a specific EcucPartition in the Com Proxy . As consequence, the Com Proxy has to map each ComUserSignal / ComUserSignalGroup via ComUserSystemTemplateSystemSignalRef/ ComUserSystemTemplateSignalGroupRef to an ComSignal resp. ComSignalGroup.

The Com module shall provide its APIs for transmission/reception requests of the relevant ComSignals and/or ComSignalGroups on the EcucPartition configured in the Com Proxy.

The Com Proxy as ComUserModule shall provide a compatible configuration structure and content for the RTE in the Application Software Cluster. It derives its configuration of ComSignals and/or ComSignalGroups from the Com. For the EcucPartition assignment, the Com Proxy creates virtual main functions (Rx/Tx) and maps the respective ComIPdus to them. These main functions exist only in the configuration but do not have an implementation.

Even if the concept of Com user provides a lot of flexibility to support access by multiple users including their notifications, some limitations have to be considered.

In general, multiple writers can cause race conditions, if the writers are not coordinated. In addition, the required sequence of Com_SendSignal (of group signals) and Com_SendSignalGroup calls cause the risk of data inconsistency in case of preemption by multiple writers.

Since such a coordination across different Software Clusters is hardly achievable, the following restriction applies:

[SWS_SwCluC_CONSTR_02653]

Upstream requirements: SRS_SwCluC_00211

⌈Sent signals or signal groups shall be provided to at most one Application Software Cluster.⌋

Nevertheless, reading a signal or signal group by several Application Software Clusters is possible.

7.4.3.3.2.1 General requirements

[SWS_SwCluC_02602]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support that the ComHandleId of ComSignals configured in Com of the Host Software Cluster can change, without reconfiguration or rebuild of the Application Software Cluster using these ComSignals.⌋

[SWS_SwCluC_02603]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support that the ComHandleId of ComSignalGroups resp. ComGroupSignals configured in the Com of the Host Software Cluster can change, without reconfiguration or rebuild of the Application Software Cluster using these ComSignalGroups.⌋

[SWS_SwCluC_02652]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support that the ComUserCbkHandleId of ComUserSignals resp. ComUserSignalGroups configured in Com users of the Application Software Cluster can change, without reconfiguration or rebuild of the Host Software Cluster.⌋

[SWS_SwCluC_02639]

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_CbkRxAck> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]), where the respective ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_ACK) the configured <ComUser_CbkRxAck> callback shall be invoked when the associated SwCluC_ComProxy_CbkRxAck instance of the Com Low Proxy is called.⌋

[SWS_SwCluC_02640]

Upstream requirements: SRS_SwCluC_00211

⌈If the invalid <ComUser_CbkInv> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]), where the respective ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_INV) the configured <ComUser_CbkInv> callback shall be invoked when the associated SwCluC_ComProxy_CbkInv instance of the Com Low Proxy is called.⌋

[SWS_SwCluC_02642]

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_CbkRxTOut> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]), where the respective ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_TOUT) the configured <ComUser_CbkRxTOut> callback shall be invoked when the

associated SwCluC_ComProxy_CbkRxTOut instance of the Com Low Proxy is called.⌋

[SWS_SwCluC_02643]

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_CbkTxAck> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]), where the respective ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_ACK) the configured <ComUser_CbkTxAck> callback shall be invoked when the associated instance of the Com Low Proxy is called.⌋

[SWS_SwCluC_02644]

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_CbkTxErr> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]), where the respective ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_ERR) the configured <ComUser_CbkTxErr> callback shall be invoked when the associated SwCluC_ComProxy_CbkTxErr instance of the Com Low Proxy is called.⌋

[SWS_SwCluC_02645]

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_CbkTxTOut> timeout notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]), where the respective ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_TOUT) the configured <ComUser_CbkTxTOut> callback shall be invoked when the associated SwCluC_ComProxy_CbkTxTOut instance of the Com Low Proxy is called.⌋

[SWS_SwCluC_02669] Guard value calculation for ComSignal

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall use the following attributes from an ComSignal for the guard value calculation

- ComSignalEndianness
- ComSignalType
- ComSignalLength
- ComBitSize

⌋

[SWS_SwCluC_02670] Guard value calculation for ComSignalGroup

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall use the following attributes from an ComSignalGroup for the guard value calculation

- ComSignalGroupArrayAccess
- From any ComGroupSignal of a ComSignalGroup
  - ComSignalEndianness
  - ComSignalType
  - ComSignalLength
  - ComBitSize

⌋

7.4.3.3.2.2 Com High Proxy

[SWS_SwCluC_02607]

Upstream requirements: SRS_SwCluC_00211

⌈The Com High Proxy shall provide the symbolic name values for the ComSignals, for Com users in the Application Software Cluster.⌋

[SWS_SwCluC_02608]

Upstream requirements: SRS_SwCluC_00211

⌈The Com High Proxy shall provide the symbolic name values for the ComSignalGroups resp. ComGroupSignals, for Com users in the Application Software Cluster.⌋

[SWS_SwCluC_02609]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall provide the functions for transmission requests of ComSignals and ComSignalGroups

- Com_SendSignal
- Com_SendDynSignal
- Com_SendSignalGroup
- Com_SendSignalGroupArray
- Com_SendSignalWithMetaData
- Com_SendDynSignalWithMetaData
- Com_SendSignalGroupWithMetaData
- Com_SendSignalGroupArrayWithMetaData

required by the Com users in the Application Software Cluster.

⌋

[SWS_SwCluC_02665] Serialization in Com_SendSignalGroup

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The function Com_SendSignalGroup provided by the Com High Proxy shall

- perform a transformation from the original data passed from the RTE in Application Software Cluster to the uint8 array,

  AND

- invoke the Com_SendSignalGroupArray API to send the data as uint8 array representation.

⌋

[SWS_SwCluC_02666] Serialization in Com_SendSignalGroupWithMetaData

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The function Com_SendSignalGroupWithMetaData provided by the Com High Proxy shall

- perform a transformation from the original data passed from the RTE in Application Software Cluster to the uint8 array,

  AND

- invoke the Com_SendSignalGroupArrayWithMetaData API to send the data as uint8 array representation.

⌋

[SWS_SwCluC_02610]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall provide the functions for reception requests of a ComSignal and ComSignalGroup

- Com_ReceiveSignal
- Com_ReceiveDynSignal
- Com_ReceiveSignalGroup
- Com_ReceiveSignalGroupArray
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_ReceiveSignalGroupWithMetaData
- Com_ReceiveSignalGroupArrayWithMetaData

required by the Com users in the Application Software Cluster

⌋

[SWS_SwCluC_02648]

Upstream requirements: SRS_SwCluC_00211

⌈The function Com_ReceiveSignalGroup provided by the Com High Proxy shall

- invoke the Com_ReceiveSignalGroupArray API to fetch the data as uint8 array representation, AND
- perform a transformation on the uint8 array to output the original data which will be passed to the RTE in the Application Software Cluster.

⌋

[SWS_SwCluC_02649]

Upstream requirements: SRS_SwCluC_00211

⌈The function Com_ReceiveSignalGroupWithMetaData provided by the Com High Proxy shall

- invoke the Com_ReceiveSignalGroupArrayWithMetaData API to fetch the data as uint8 array representation, AND
- perform a transformation on the uint8 array to output the original data which will be passed to the RTE in the Application Software Cluster.

⌋

[SWS_SwCluC_02611]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall provide the functions for invalidation requests of a ComSignal and ComSignalGroup

- Com_InvalidateSignal
- Com_InvalidateSignalGroup

required by the Com users in the Application Software Cluster

⌋

[SWS_SwCluC_02612]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202, SRS_SwCluC_00213

⌈If the Com High Proxy is not connected to the Com Low Proxy, the functions w.r.t transmission requests of ComSignals and ComSignalGroups

- Com_SendSignal
- Com_SendDynSignal
- Com_SendSignalGroup
- Com_SendSignalGroupArray

- Com_SendSignalWithMetaData
- Com_SendDynSignalWithMetaData
- Com_SendSignalGroupWithMetaData
- Com_SendSignalGroupArrayWithMetaData

of the Com High Proxy shall return COM_SERVICE_NOT_AVAILABLE.

⌋

[SWS_SwCluC_02613]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com High Proxy is not connected to the Com Low Proxy, the functions w.r.t reception requests of ComSignals and ComSignalGroups

- Com_ReceiveSignal
- Com_ReceiveDynSignal
- Com_ReceiveSignalGroup
- Com_ReceiveSignalGroupArray
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_ReceiveSignalGroupWithMetaData
- Com_ReceiveSignalGroupArrayWithMetaData
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData

of the Com High Proxy shall return COM_SERVICE_NOT_AVAILABLE.

⌋

[SWS_SwCluC_02614]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com High Proxy is not connected to the Com Low Proxy, the functions w.r.t invalidation requests of ComSignals and ComSignalGroups

- Com_InvalidateSignal
- Com_InvalidateSignalGroup

of the Com High Proxy shall return COM_SERVICE_NOT_AVAILABLE.

⌋

Callback Functions

[SWS_SwCluC_02618]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_CbkRxAck> notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_ACK.⌋

[SWS_SwCluC_02619]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional invalid <ComUser_CbkInv> notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_INV.⌋

[SWS_SwCluC_02620]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_CbkRxTOut> timeout notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_TOUT.⌋

[SWS_SwCluC_02621]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_CbkTxAck> notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_ACK.⌋

[SWS_SwCluC_02622]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_CbkTxErr> error notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_ERR.⌋

[SWS_SwCluC_02623]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_CbkTxTOut> timeout notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_TOUT.⌋

Please note as well [SWS_SwCluC_CONSTR_02667]!

[SWS_SwCluC_02647]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00212

⌈The Com High Proxy shall create for each SwCluCComProxyBaseSocket a required resource entry in the Binary Manifest.⌋

Please note that the callbacks of the Application Software Cluster

- <ComUser_CbkRxAck>
- <ComUser_CbkInv>
- <ComUser_CbkRxTOut>
- <ComUser_CbkTxAck>
- <ComUser_CbkTxErr>
- <ComUser_CbkTxTOut>

are connected to the Host Software Cluster via the Com base socket. See [SWS_SwCluC_02661]

In the Application Software Cluster, the relationship between a Signal or a Signal Group in Com and the related Com base socket is given indirectly, via the common EcucPartition(s) of the SwCluCComProxyBaseSocket.SwCluCProxyEcucPartitionRef and the ComMainFunctionRx. ComMainRxPartitionRef and ComMainFunctionTx.ComMainTxPartitionRef respectively. Those relationships are illustrated in figure 7.16.

The reason not to define a dedicated EcuC configuration structure for this purpose, is the assumption that the same tools are used to derive the Com configuration for an Application Software Cluster and a Host Software Cluster.

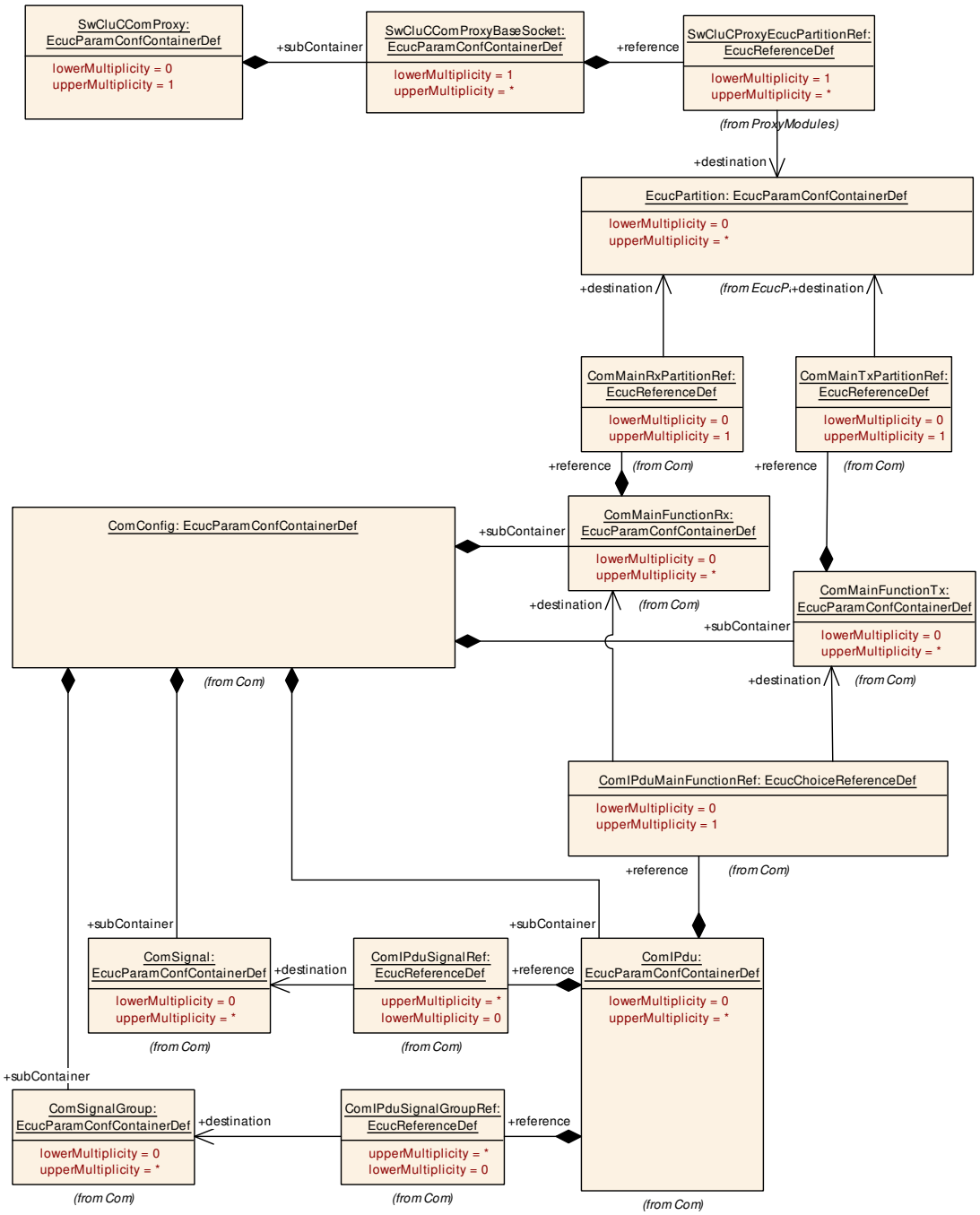

Figure 7.16: Signal or a Signal Group and their relation ships to Com base sockets via EcucPartition

[SWS_SwCluC_02662]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00212

⌈The `Com High Proxy` shall create for each `ComSignal` and `ComSignalGroup` configured in Com a required resource entry in the `Binary Manifest`. Where the `ComMainFunctionRx.ComMainRxPartitionRef` and `ComMainFunctionTx.ComMainTxPartitionRef` are relevant to denote the `EcucPartition`.⌋

7.4.3.3.2.3 Com Low Proxy

[SWS_SwCluC_02651]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide in its ComUserModuleCnf containers the ComUserCbkHandleId values for the configured ComUserSignals and ComUserSignalGroups, according to its internal implementation requirements (e.g. to address the Com Low Proxy's data structures).⌋

[SWS_SwCluC_02625]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide the entry function for transmission, reception and invalidation requests of ComSignal(Group)s, according to [SWS_SwCluC_02609], [SWS_SwCluC_02610] and [SWS_SwCluC_02611], once per configured SwCluCComProxyBaseSocket, at the EcucPartition configured by SwCluCComProxyBaseSocket.SwCluCProxyEcucPartitionRef.
In case the SwCluCComProxyUsedSatelliteRef is set, the EcucPartition shall be changed, before the Com API is called.⌋

[SWS_SwCluC_02627]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_CbkRxAck notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_RX_ACK.⌋

[SWS_SwCluC_CONSTR_02655]

Upstream requirements: SRS_SwCluC_00211

⌈Each notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_ACK, shall be configured as SwCluC_ComProxy_<BS>_CbkRxAck, where <BS> is the shortName of the SwCluCComProxyBaseSocket container⌋

[SWS_SwCluC_02628]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_CbkInv notification callbacks, configured on each ComSignal(Group) via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_RX_INV.⌋

[SWS_SwCluC_CONSTR_02656]

Upstream requirements: SRS_SwCluC_00211

⌈Each notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_INV, shall be configured as

SwCluC_ComProxy_<BS>_CbkInv, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_02629]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_CbkRxTOut timeout notification callbacks configured on each ComSignal(Group), via the ComUserCallbackName (See [ECUC_Com_10036]), with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_RX_TOUT.⌋

[SWS_SwCluC_CONSTR_02657]

Upstream requirements: SRS_SwCluC_00211

⌈Each notification callback where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_TOUT shall be configured as SwCluC_ComProxy_<BS>_CbkRxTOut where <BS> is the shortName of the SwCluCComProxyBaseSocket container⌋

[SWS_SwCluC_02630]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_CbkTxAck notification callbacks configured on each ComSignal(Group), via the ComUserCallbackName (See [ECUC_Com_10036]), with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TX_ACK.⌋

[SWS_SwCluC_CONSTR_02658]

Upstream requirements: SRS_SwCluC_00211

⌈Each notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_ACK, shall be configured as SwCluC_ComProxy_<BS>_CbkTxAck, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_02631]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_CbkTxErr notification callbacks configured on each ComSignal(Group), via the ComUserCallbackName (See [ECUC_Com_10036]), with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TX_ERR.⌋

[SWS_SwCluC_CONSTR_02659]

Upstream requirements: SRS_SwCluC_00211

⌈Each notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_ERR, shall be configured as SwCluC_ComProxy_<BS>_CbkTxErr, where <BS> is the shortName of the SwCluCComProxyBaseSocket container⌋

[SWS_SwCluC_02632]

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_CbkTxTOut notification callbacks configured on each ComSignal(Group), via the ComUserCallbackName (See [ECUC_Com_10036]), with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TX_TOUT.⌋

[SWS_SwCluC_CONSTR_02660]

Upstream requirements: SRS_SwCluC_00211

⌈Each notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_TOUT, shall be configured as SwCluC_ComProxy_<BS>_CbkTxTOut, where <BS> is the shortName of the SwCluCComProxyBaseSocket container⌋

[SWS_SwCluC_02633]

Upstream requirements: SRS_SwCluC_00211

⌈The notification callback instances according to [SWS_SwCluC_02626], [SWS_SwCluC_02627], [SWS_SwCluC_02628], [SWS_SwCluC_02629], [SWS_SwCluC_02630], [SWS_SwCluC_02631] and [SWS_SwCluC_02632] are provided

- at the EcucPartition of the SwCluCComProxyBaseSocket referenced by the SwCluCComProxyBaseSocket - via SwCluCComProxyUsedSatelliteRef. In this case, the EcucPartition shall be changed, before the callback in the High Proxy is called.

⌋

[SWS_SwCluC_02646]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00212

⌈The Com Low Proxy shall create for each SwCluCComProxyBaseSocket a provided resource entry in the Binary Manifest.

⌋

[SWS_SwCluC_02661]

Upstream requirements: SRS_SwCluC_00211

⌈The Com base socket shall implement the connection of the related callback functions

- SwCluC_ComProxy_CbkRxAck
- SwCluC_ComProxy_CbkInv
- SwCluC_ComProxy_CbkRxTOut
- SwCluC_ComProxy_CbkTxAck
- SwCluC_ComProxy_CbkTxErr

- SwCluC_ComProxy_CbkTxTOut

⌋

[SWS_SwCluC_02634]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com Low Proxy is not connected to the Com High Proxy, the callback functions

- SwCluC_ComProxy_CbkRxAck
- SwCluC_ComProxy_CbkInv
- SwCluC_ComProxy_CbkRxTOut
- SwCluC_ComProxy_CbkTxAck
- SwCluC_ComProxy_CbkTxErr
- SwCluC_ComProxy_CbkTxTOut

of the Com Low Proxy shall return shall return without effect.⌋

Conceptually, the Com Proxy provides Signals and Signal Groups with their corresponding notifications for a set of specific EcucPartitions. In the simplest case, this is a single EcucPartition, where also the Com module handles the underlying ComIPdu. In case the Com users as well as the Com API are unspecific to the EcucPartition, the Com Proxy can provide the same Signals and Signal Groups with their corresponding notifications for more than one EcucPartition. Nevertheless, it's possible that the Com Proxy changes the partition for Com API access and notifications. But be aware that this may have a severe performance impact, if this is used for a larger number of Signals or Signal Groups.

In the Host Software Cluster, the set of Signals and Signal Groups, with their corresponding notifications, is configured by means of the SwCluCComProxyBaseSocket.ComUserModuleCnf. For those Signals and Signal Groups, the related entries in the Binary Manifest are created by the Com Low Proxy.

[SWS_SwCluC_02663]

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00212

⌈The Com Low Proxy shall create for each ComUserSignal and ComUserSignalGroup configured in a SwCluCComProxyBaseSocket.ComUserModuleCnf a provided resource entry in the Binary Manifest.⌋

7.4.3.3.3 ComIPdu based proxy handling

If using a data path in Com that refers to ComIPdus, then a ComUserModuleCnf is required for each EcucPartition, in which the Com Proxy

- requires direct access to the Com APIs initiating transmission requests
- provides notification callbacks w.r.t transmission and reception

Effectively, a SwCluCComProxyBaseSocket links a fixed set of notification callbacks in the Com to a specific EcucPartition in the Com Proxy. As a consequence, the Com Proxy has to map each ComUserIPdu via ComUserSystemTemplate-SystemSignalRef to an ComIPdu.

Even if the concept of Com user provides a lot of flexibility to support access by multiple users, including their notifications, some limitations need to be considered. In general, multiple writers can cause race conditions, if the writers are not coordinated. In addition, the behavior of <ComUser_ComCbkTriggerTransmit> does not support multiple notified users for the same ComIPdu. Since such a coordination across different Software Clusters is hardly achievable, the following restriction applies:

[SWS_SwCluC_CONSTR_03546] Multiplicity of provided ComIPdus

Upstream requirements: SRS_SwCluC_00211

⌈Sent ComIPdu shall be provided to at most one Application Software Cluster.⌋

In addition, the Com Proxy does not support a fan out of ComIPdus sent or received via transport protocol API.

Please note that the PduR supports a fan-out of TP-connections to upper layers.

7.4.3.3.3.1 General requirements

[SWS_SwCluC_03502] Support of changing Host Software Cluster without rebuilding Application Software Cluster

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support that the ComIPdu Ids of the Com of the Host Software Cluster can change, without reconfiguration or rebuild of the Application Software Cluster using these ComIPdus.⌋

[SWS_SwCluC_03548] Support of changing Application Software Cluster without rebuilding Host Software Cluster

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall support that the ComUserCbkHandleId of ComUserIPdus configured in Com users of the Application Software Cluster can change, without reconfiguration or rebuild of the Host Software Cluster.⌋

[SWS_SwCluC_03535] Invoke all the configured <ComUser_ComCbkRx-Indication> if SwCluC_ComProxy_ComCbkRxIndication is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkRxIndication> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_INDICATION), all the configured <ComUser_ComCbkRxIndication> callbacks shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkRxIndication from the Com Low Proxy is called.⌋

[SWS_SwCluC_03536] Invoke the configured <ComUser_ComCbkTx-Confirmation> if SwCluC_ComProxy_ComCbkRxIndication is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkTxConfirmation> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_CONFIRMATION), the configured <ComUser_ComCbkTxConfirmation> callback shall be invoked. when the associated instance of the SwCluC_ComProxy_ComCbkTxConfirmation from the Com Low Proxy is called.⌋

[SWS_SwCluC_03537] Invoke the configured <ComUser_ComCbkTrigger-Transmit> if SwCluC_ComProxy_ComCbkTriggerTransmit is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkTriggerTransmit> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_TRIGGER_TRANSMIT), the configured <ComUser_ComCbkTrigger-Transmit> callback shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkTriggerTransmit from the Com Low Proxy is called.⌋

[SWS_SwCluC_03538] Invoke the configured <ComUser_ComCbkCopyRxData> if SwCluC_ComProxy_ComCbkCopyRxData is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkCopyRxData> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_RX_DATA), the configured <ComUser_ComCbkCopyRxData> callback shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkCopyRx-Data from the Com Low Proxy is called.⌋

[SWS_SwCluC_03539] Invoke the configured <ComUser_ComCbkStartOfReception> if SwCluC_ComProxy_ComCbkStartOfReception is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkStartOfReception> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_START_OF_RECEPTION), the configured <ComUser_ComCbkStartOfReception> callback shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkStartOfReception from the Com Low Proxy is called.⌋

[SWS_SwCluC_03540] Invoke the configured <ComUser_ComCbkTpRxIndication> if SwCluC_ComProxy_ComCbkTpRxIndication is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkTpRxIndication> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_RX_INDICATION), the configured <ComUser_ComCbkTpRxIndication> callback shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkTpRxIndication from the Com Low Proxy is called.⌋

[SWS_SwCluC_03541] Invoke the configured <ComUser_ComCbkTpTxConfirmation> if SwCluC_ComProxy_ComCbkTpTxConfirmation is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkTpTxConfirmation> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_TX_CONFIRMATION), the configured <ComUser_ComCbkTpTxConfirmation> callback shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkTpTxConfirmation from the Com Low Proxy is called.⌋

[SWS_SwCluC_03542] Invoke the configured <ComUser_ComCbkCopyTxData> if SwCluC_ComProxy_ComCbkCopyTxData is called

Upstream requirements: SRS_SwCluC_00211

⌈If the <ComUser_ComCbkCopyTxData> notification callback is configured in the Com High Proxy (via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_TX_DATA), the configured <ComUser_ComCbkCopyTxData> callback shall be invoked, when the associated instance of the SwCluC_ComProxy_ComCbkCopyTxData from the Com Low Proxy is called.⌋

[SWS_SwCluC_03554] Guard value calculation for ComIPdu

Upstream requirements: SRS_SwCluC_00211

⌈The Com Proxy shall use the following attributes from an ComIPdu for the guard value calculation

- ComIPduDirection
- ComIPduType

⌋

7.4.3.3.3.2 Com High Proxy

[SWS_SwCluC_03506] The Com High Proxy shall provide the symbolic name values for the ComIPdu IDs

Upstream requirements: SRS_SwCluC_00211

⌈The Com High Proxy shall provide the symbolic name values for the ComIPdu IDs, for Com users in the Application Software Cluster.⌋

Note: According parameters are called TxPduId in Com_Transmit and id in the callback function for Large Data Path in Com.

[SWS_SwCluC_03507] Com High Proxy shall provide Com_Transmit

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall provide the function w.r.t transmission request of a ComIPdu

- Com_Transmit

required by the Application Software Cluster

⌋

[SWS_SwCluC_03508] Behaviour if Com_Transmit is called and Com High Proxy ia not connected to Com Low Proxy

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com High Proxy is not connected to the Com Low Proxy, the function

- Com_Transmit

of the Com High Proxy shall return E_NOT_OK.⌋

Callback Functions

[SWS_SwCluC_03510] Support of <ComUser_ComCbkRxIndication> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the <ComUser_ComCbkRxIndication> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_INDICATION⌋

[SWS_SwCluC_03511] Support of optional <ComUser_ComCbkTxConfirmation> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_ComCbkTxConfirmation> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_CONFIRMATION⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkTxConfirmation>.

[SWS_SwCluC_03512] Support the optional <ComUser_ComCbkTriggerTransmit> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the optional <ComUser_ComCbkTriggerTransmit> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_TRIGGER_TRANSMIT⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkTriggerTransmit>.

[SWS_SwCluC_03513] Support the <ComUser_ComCbkCopyRxData> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the <ComUser_ComCbkCopyRxData> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_RX_DATA⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkCopyRxData>.

[SWS_SwCluC_03514] Support of <ComUser_ComCbkStartOfReception> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the <ComUser_ComCbkStartOfReception> notification callback configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_RX_START_OF_RECEPTION⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkStartOfReception>.

[SWS_SwCluC_03515] Support the <ComUser_ComCbkTpRxIndication> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the <ComUser_ComCbkTpRxIndication> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_RX_INDICATION⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkTpRxIndication>.

[SWS_SwCluC_03516] Support of <ComUser_ComCbkTpTxConfirmation> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the <ComUser_ComCbkTpTxConfirmation> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_TX_CONFIRMATION⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkTpTxConfirmation>.

[SWS_SwCluC_03517] Support of <ComUser_ComCbkCopyTxData> notification callback

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00202

⌈The Com High Proxy shall support the <ComUser_ComCbkCopyTxData> notification callback, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, where the ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_TX_DATA⌋

Please note that only a single Com user can be notified with <ComUser_ComCbkCopyTxData>.

Please note as well [SWS_SwCluC_CONSTR_02667]!

[SWS_SwCluC_03543] Create a required resource entry in the Binary Manifest

Upstream requirements: SRS_SwCluC_00211

⌈The Com High Proxy shall create, for each SwCluCComProxyBaseSocket, a required resource entry in the Binary Manifest.⌋

Please note that the callbacks of the Application Software Cluster

- <ComUser_ComCbkStartOfReception>
- <ComUser_ComCbkCopyRxData>
- <ComUser_ComCbkCopyTxData>
- <ComUser_ComCbkTriggerTransmit>
- <ComUser_ComCbkRxIndication>
- <ComUser_ComCbkTpRxIndication>
- <ComUser_ComCbkTxConfirmation>
- <ComUser_ComCbkTpTxConfirmation>

are connected to the Host Software Cluster via the Com base socket. See [SWS_SwCluC_03550]

In the Application Software Cluster, the relationship between a ComUserIPdu and the related Base Socket is given indirectly, via the common EcucPartition(s) of the SwCluCComProxyBaseSocket.SwCluCProxyEcucPartitionRef and the EcucPartition of the ComIPdu's referenced Pdu. Where the EcucPartition of the Pdu is given via Pdu.EcucPduDefaultPartitionRef or Pdu.EcucPduDedicatedPartition.EcucPduDedicatedPartitionRef, where EcucPduDedicatedPartition applies for Com configuration. Those relationships are illustrated in figure 7.16. The reason not to define a dedicated EcuC configuration structure for this purpose, is the assumption that the same tools are used to derive the Com configuration for an Application Software Cluster and a Host Software Cluster.

Those relationships are illustrated in figure 7.16.

[SWS_SwCluC_03551] Create for each ComIPdu configured in Com a required resource entry in the Binary Manifest

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00212

⌈The Com High Proxy shall create for each ComIPdu configured in Com a required resource entry in the Binary Manifest. Where the EcucPartition of the ComIPdu's Pdu given via Pdu.EcucPduDefaultPartitionRef or Pdu.EcucPduDedicatedPartition.EcucPduDedicatedPartitionRef are relevant to denote the EcucPartition.⌋

7.4.3.3.3.3 Com Low Proxy

[SWS_SwCluC_03549] Provision of `ComUserCbkHandleId` values

Upstream requirements: SRS_SwCluC_00211

⌈The `Com Low Proxy` shall provide in its `ComUserModuleCnf` containers the `ComUserCbkHandleId` values for the configured `ComUserIPdu`s, according to its internal implementation requirements (e.g. to address the `Com Low Proxy`'s data structures).⌋

[SWS_SwCluC_03519] The `Com Low Proxy` shall provide the entry function for transmission requests

Upstream requirements: SRS_SwCluC_00211

⌈The `Com Low Proxy` shall provide the entry function for transmission requests, according to [SWS_SwCluC_03506], once per configured `SwCluCComProxyBaseSocket` at the `EcucPartition` configured by `SwCluCComProxyBaseSocket.SwCluCProxyEcucPartitionRef`.
⌋

[SWS_SwCluC_03521] Provide an instance of all `SwCluC_ComProxy_ComCbkRxIndication` notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The `Com Low Proxy` shall provide an instance of all `SwCluC_ComProxy_ComCbkRxIndication` notification callbacks, configured via the `ComUserCallbackName` (See [ECUC_Com_10036]) parameter, with `ComUserCallbackType` (See [ECUC_Com_10034]) set to `COM_RX_INDICATION`.⌋

[SWS_SwCluC_CONSTR_03555] Existence of `SwCluC_ComProxy_ComCbkRxIndication`

Upstream requirements: SRS_SwCluC_00211

⌈Each `SwCluC_ComProxy_ComCbkRxIndication` notification callback, where the parameter `ComUserCallbackType` (See [ECUC_Com_10034]) is set to `COM_RX_INDICATION`, shall be configured as `SwCluC_ComProxy_<BS>_ComCbkRxIndication`, where `<BS>` is the `shortName` of the `SwCluCComProxyBaseSocket` container.⌋

[SWS_SwCluC_03522] Provide an instance of all `SwCluC_ComProxy_ComCbkTxConfirmation` notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The `Com Low Proxy` shall provide an instance of all `SwCluC_ComProxy_ComCbkTxConfirmation` notification callbacks, configured via the `ComUserCallbackName` (See [ECUC_Com_10036]) parameter, with `ComUserCallbackType` (See [ECUC_Com_10034]) set to `COM_TX_CONFIRMATION`.⌋

[SWS_SwCluC_CONSTR_03556] Existence of SwCluC_ComProxy_ComCbkTxConfirmation

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkTxConfirmation notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_CONFIRMATION, shall be configured as SwCluC_ComProxy_<BS>_ComCbkTxConfirmation, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03523] Provide an instance of all SwCluC_ComProxy_ComCbkTriggerTransmit notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_ComCbkTriggerTransmit notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TX_TRIGGER_TRANSMIT.⌋

[SWS_SwCluC_CONSTR_03557] Existence of SwCluC_ComProxy_ComCbkTriggerTransmit

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkTriggerTransmit notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TX_TRIGGER_TRANSMIT, shall be configured as SwCluC_ComProxy_<BS>_ComCbkTriggerTransmit, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03524] Provide an instance of all SwCluC_ComProxy_ComCbkCopyRxData notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_ComCbkCopyRxData notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TP_COPY_RX_DATA.⌋

[SWS_SwCluC_CONSTR_03558] Existence of SwCluC_ComProxy_ComCbkCopyRxData

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkCopyRxData notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_RX_DATA, shall be configured as SwCluC_ComProxy_<BS>_ComCbkCopyRxData, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03525] Provide an instance of all SwCluC_ComProxy_ComCbkStartOfReception notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_ComCbkStartOfReception notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with ComUserCallbackType (See [ECUC_Com_10034]) set to COM_RX_START_OF_RECEPTION.⌋

[SWS_SwCluC_CONSTR_03559] Existence of SwCluC_ComProxy_ComCbkStartOfReception

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkStartOfReception notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_RX_DATA, shall be configured as SwCluC_ComProxy_<BS>_ComCbkCopyRxData, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03526] Provide an instance of all SwCluC_ComProxy_ComCbkTpRxIndication notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_ComCbkTpRxIndication notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TP_RX_INDICATION.⌋

[SWS_SwCluC_CONSTR_03560] Existence of SwCluC_ComProxy_ComCbkTpRxIndication

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkTpRxIndication notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_RX_INDICATION, shall be configured as SwCluC_ComProxy_<BS>_ComCbkTpRxIndication, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03527] Provide an instance of all SwCluC_ComProxy_ComCbkTpTxConfirmation notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_ComCbkTpTxConfirmation notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TP_TX_CONFIRMATION.⌋

[SWS_SwCluC_CONSTR_03561] Existence of SwCluC_ComProxy_ComCbkTpTxConfirmation

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkTpTxConfirmation notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_TX_CONFIRMATION, shall be configured as SwCluC_ComProxy_<BS>_ComCbkTpTxConfirmation, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03528] Provide an instance of all SwCluC_ComProxy_ComCbkCopyTxData notification callbacks

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall provide an instance of all SwCluC_ComProxy_ComCbkCopyTxData notification callbacks, configured via the ComUserCallbackName (See [ECUC_Com_10036]) parameter, with the ComUserCallbackType (See [ECUC_Com_10034]) set to COM_TP_COPY_TX_DATA.⌋

[SWS_SwCluC_CONSTR_03562] Existence of SwCluC_ComProxy_ComCbkCopyTxData

Upstream requirements: SRS_SwCluC_00211

⌈Each SwCluC_ComProxy_ComCbkCopyTxData notification callback, where the parameter ComUserCallbackType (See [ECUC_Com_10034]) is set to COM_TP_COPY_TX_DATA, shall be configured as SwCluC_ComProxy_<BS>_ComCbkCopyTxData, where <BS> is the shortName of the SwCluCComProxyBaseSocket container.⌋

[SWS_SwCluC_03529] Provision of callback instances at at the EcucPartition of the SwCluCComProxyBaseSocket

Upstream requirements: SRS_SwCluC_00211

⌈The notification callback instances according to [SWS_SwCluC_02626], [SWS_SwCluC_03521], [SWS_SwCluC_03522], [SWS_SwCluC_03523], [SWS_SwCluC_03524], [SWS_SwCluC_03525], [SWS_SwCluC_03526], [SWS_SwCluC_03527] and [SWS_SwCluC_03528] are provided

- at the EcucPartition of the SwCluCComProxyBaseSocket, referenced by the SwCluCComProxyBaseSocket - via SwCluCComProxyUsedSatelliteRef.

⌋

[SWS_SwCluC_03544] Create for each SwCluCComProxyBaseSocket a provided resource entry in the Binary Manifest

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00212

⌈The Com Low Proxy shall create for each SwCluCComProxyBaseSocket a provided resource entry in the Binary Manifest.⌋

[SWS_SwCluC_03550] The Com base socket shall implement the connection of the related callback functions

Upstream requirements: SRS_SwCluC_00211

⌈The Com base socket shall implement the connection of the related callback functions

- SwCluC_ComProxy_ComCbkStartOfReception
- SwCluC_ComProxy_ComCbkCopyRxData
- SwCluC_ComProxy_ComCbkCopyTxData
- SwCluC_ComProxy_ComCbkTriggerTransmit
- SwCluC_ComProxy_ComCbkRxIndication
- SwCluC_ComProxy_ComCbkTpRxIndication
- SwCluC_ComProxy_ComCbkTxConfirmation
- SwCluC_ComProxy_ComCbkTpTxConfirmation

⌋

[SWS_SwCluC_03530] Handling for notification callback functions if Com Low Proxy is not connected to the Com High Proxy

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com Low Proxy is not connected to the Com High Proxy, the notification callback functions

- SwCluC_ComProxy_ComCbkStartOfReception
- SwCluC_ComProxy_ComCbkCopyRxData
- SwCluC_ComProxy_ComCbkCopyTxData

of the Com Low Proxy shall return BUFREQ_E_NOT_OK.⌋

[SWS_SwCluC_03531] Handling for SwCluC_ComProxy_ComCbkTriggerTransmit If the Com Low Proxy is not connected to the Com High Proxy

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com Low Proxy is not connected to the Com High Proxy, the notification callback function

- SwCluC_ComProxy_ComCbkTriggerTransmit

of the Com Low Proxy shall return E_NOT_OK.⌋

[SWS_SwCluC_03532] Handling for notification callback functions which do not have a return value if Com Low Proxy is not connected to the Com High Proxy

Upstream requirements: SRS_SwCluC_00211, SRS_SwCluC_00213

⌈If the Com Low Proxy is not connected to the Com High Proxy, the notification callback functions

- SwCluC_ComProxy_ComCbkRxIndication
- SwCluC_ComProxy_ComCbkTpRxIndication
- SwCluC_ComProxy_ComCbkTxConfirmation
- SwCluC_ComProxy_ComCbkTpTxConfirmation

of the Com Low Proxy shall return without effect.⌋

Conceptually, the Com Proxy provides ComIPdus with their corresponding notifications for a set of specific EcucPartitions. In the simplest case, this is a single EcucPartition, where also the Com module handles the underlying ComIPdu. In case the Com users as well as the Com API are unspecific to the EcucPartition, the Com Proxy provides the same ComIPdu on multiple EcucPartitions. Nevertheless, it is possible that the Com Proxy changes the partition. But be aware that this may have a severe performance impact, if this is used for a larger number of ComIPdus.

In the Host Software Cluster, the set of ComIPdus with their corresponding notifications is configured by means of the SwCluCComProxyBaseSocket.ComUserModuleCnf. For those ComIPdus, the related entries in the Binary Manifest are created by the Com Low Proxy.

[SWS_SwCluC_03552] Com Low Proxy shall create a provided resource entry in the Binary Manifest

Upstream requirements: SRS_SwCluC_00211

⌈The Com Low Proxy shall create, for each ComUserIPdu configured in a SwCluCComProxyBaseSocket.ComUserModuleCnf, a provided resource entry in the Binary Manifest.⌋

7.4.3.3.4 Error Detection

[SWS_SwCluC_02636]

Upstream requirements: SRS_SwCluC_00211, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If the development error detection is enabled for the Com High Proxy, the APIs

- Com_SendSignal
- Com_SendDynSignal

- Com_SendSignalGroup
- Com_SendSignalGroupArray
- Com_SendSignalWithMetaData
- Com_SendDynSignalWithMetaData
- Com_SendSignalGroupWithMetaData
- Com_SendSignalGroupArrayWithMetaData
- Com_ReceiveSignal
- Com_ReceiveDynSignal
- Com_ReceiveSignalGroup
- Com_ReceiveSignalGroupArray
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_ReceiveSignalGroupWithMetaData
- Com_ReceiveSignalGroupArrayWithMetaData
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_InvalidateSignal
- Com_InvalidateSignalGroup

shall report the DET error COM_E_UNINIT, in case the Com High Proxy is not initialized.⌋

[SWS_SwCluC_02637]

Upstream requirements: SRS_SwCluC_00211, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If the development error detection is enabled for the Com High Proxy, the APIs

- Com_SendSignal
- Com_SendDynSignal
- Com_SendSignalGroup
- Com_SendSignalGroupArray
- Com_SendSignalWithMetaData
- Com_SendDynSignalWithMetaData
- Com_SendSignalGroupWithMetaData

- Com_SendSignalGroupArrayWithMetaData
- Com_ReceiveSignal
- Com_ReceiveDynSignal
- Com_ReceiveSignalGroup
- Com_ReceiveSignalGroupArray
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_ReceiveSignalGroupWithMetaData
- Com_ReceiveSignalGroupArrayWithMetaData
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_InvalidateSignal
- Com_InvalidateSignalGroup

shall, when invoked with a NULL_PTR, report the DET error COM_E_PARAM_POINTER.⌋

[SWS_SwCluC_02638]

Upstream requirements: SRS_SwCluC_00211, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If the development error detection is enabled for the Com High Proxy, the APIs

- Com_SendSignal
- Com_SendDynSignal
- Com_SendSignalGroup
- Com_SendSignalGroupArray
- Com_SendSignalWithMetaData
- Com_SendDynSignalWithMetaData
- Com_SendSignalGroupWithMetaData
- Com_SendSignalGroupArrayWithMetaData
- Com_ReceiveSignal
- Com_ReceiveDynSignal
- Com_ReceiveSignalGroup
- Com_ReceiveSignalGroupArray

- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_ReceiveSignalGroupWithMetaData
- Com_ReceiveSignalGroupArrayWithMetaData
- Com_ReceiveSignalWithMetaData
- Com_ReceiveDynSignalWithMetaData
- Com_InvalidateSignal
- Com_InvalidateSignalGroup

shall, when invoked with a wrong parameter, report the DET error COM_E_PARAM.⌋

[SWS_SwCluC_03533] DET error handling if Com_Transmit is called for an uninitialized Com High Proxy

Upstream requirements: SRS_SwCluC_00211, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If the development error detection is enabled for the Com High Proxy, the API

- Com_Transmit

shall report the DET error COM_E_UNINIT, in case the Com High Proxy is not initialized.⌋

[SWS_SwCluC_03534] DET error handling if Com_Transmit is invoked with a NULL_PTR

Upstream requirements: SRS_SwCluC_00211, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If the development error detection is enabled for the Com High Proxy, the API

- Com_Transmit

shall, when invoked with a NULL_PTR, report the DET error COM_E_PARAM_POINTER.

⌋

7.4.3.4 Dcm Proxy

The specified Dcm Proxy has the following underlying design principle:

For diagnostic services accessing data of the ECU's applications, the Dcm requires interfaces to the software components, to read or write data values or to control specific functionality. Those interfaces need to be provided by the software components that are implementing the regular functionality of the ECU. Therefore, the Dcm Proxy provides the functionality to route these interfaces into the Application Software Cluster. The protocol, session and authentication handling stay in the Dcm of the

Host Software Cluster. The configuration whether a sub-service is available at all, including its properties (e.g. a data size or the required diagnostic session), is still located in the Dcm.

For some diagnostic services, the Dcm offers a wide configurability with regard to how the interface to the software components is exactly implemented. E.g., the access to a diagnostic data can be done either with a client server interface for synchronous access, or a client server interface for asynchronous access, or a sender receiver interface. Furthermore, additional sub-options are possible.

To avoid configuration dependencies from the Host Software Cluster to detailed interface decisions in the Application Software Clusters, the Dcm Proxy uses a more generic interface to connect to the Dcm. The configuration of interface specialization is done in the Application Software Clusters.

In general, this specification refers to the Standardized AUTOSAR Interfaces of the Dcm

7.4.3.4.1 Enable Dcm Proxy Generation

[SWS_SwCluC_02338]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00204

⌈The Dcm High Proxy code, and related AUTOSAR model descriptions, shall only be created if the configuration parameter SwCluCProxyGenerationDcm is set to HIGH_PROXY.⌋

[SWS_SwCluC_02339]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00204

⌈The Dcm Low Proxy code, and related AUTOSAR model descriptions, shall only be created if the configuration parameter SwCluCProxyGenerationDcm is set to LOW_PROXY.⌋

7.4.3.4.2 Dcm High Proxy

[SWS_SwCluC_02300]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm High Proxy shall provide the header file Dcm.h.⌋

Multiple Partitions

[SWS_SwCluC_02341]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm High Proxy shall create one partition specific Service Software Component per configured EcucPartition, with the name Dcm_<EcucPartition shortName>.⌋

[SWS_SwCluC_02342]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm High Proxy shall provide the Ports belonging to a specific interface connection at the partition specific Service Software Component, at the given EcucPartition of the proxy channel, configured via

- SwCluCDcmProxyDiagnosticData.SwCluCProxyEcucPartitionRef

  OR

- SwCluCDcmProxyRoutineControl.SwCluCProxyEcucPartitionRef

  OR

- SwCluCDcmProxyServiceRequestNotification.SwCluCProxyEcucPartitionRef

⌋

Binary Manifest

The implementation of the Dcm High Proxy may require some basic infrastructure from the Dcm Low Proxy in order to be operational. Such generic infrastructure services can be linked between Application Software Cluster and Host Software Cluster by means of a Dcm base socket.

[SWS_SwCluC_02390]

Upstream requirements: SRS_SwCluC_00212

⌈The Dcm High Proxy shall create for each SwCluCDcmProxyBaseSocket a required resource entry in the Binary Manifest.⌋

7.4.3.4.2.1 Diagnostic Data proxy

The diagnostic data proxy provides access to diagnostic data in Application Software Clusters, which can be accessed by diagnostic services via the Dcm in the Host Software Cluster.

Service ID	Service name
UDS 0x22	ReadDataByIdentifier
UDS 0x24	ReadScalingDataByIdentifier
UDS 0x2E	WriteDataByIdentifier
UDS 0x2F	InputOutputControlByIdentifier

Service ID	Service name
OBD $01	Request Current Powertrain Diagnostic Data
OBD $09	Request Vehicle Information

Table 7.3: Diagnostic Services for diagnostic data proxy

In addition, the diagnostic data proxy is also used by the Dem to access diagnostic data for freeze frame capturing.

DataServices_{Data} ports

The DataServices_{Data} ports are used to access single diagnostic data elements.

[SWS_SwCluC_02301]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{Data} RPortPrototypes for synchronous ClientServerInterface DataServices_{Data} for a SwCluCDcmProxyDiagnosticData referencing via SwCluCDcmProxyDataRef a DcmDspData or a DcmDspPidService01.⌋

Note: In the corresponding Dcm configuration, DcmDspDataUsePort will usually be set to USE_DATA_SYNCH_CLIENT_SERVER.

[SWS_SwCluC_02302]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{Data} RPortPrototypes for asynchronous ClientServerInterface DataServices_{Data} without parameter ErrorCode.⌋

Note: In the corresponding Dcm configuration, DcmDspDataUsePort will usually be set to USE_DATA_ASYNCH_CLIENT_SERVER.

[SWS_SwCluC_02303]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{Data} RPortPrototypes for asynchronous ClientServerInterface DataServices_{Data} with parameter ErrorCode.⌋

Note: In the corresponding Dcm configuration, DcmDspDataUsePort will usually be set to USE_DATA_ASYNCH_CLIENT_SERVER_ERROR.

[SWS_SwCluC_02304]

Upstream requirements: SRS_SwCluC_00208

⌈In case the SwCluCDcmProxyDiagnosticData channel related to a DataServices_{Data} PortPrototypes is connected, the Dcm Proxy shall invoke the

- ReadData operation, when the corresponding Xxx_ReadData API is called in the Dcm Low Proxy,

  OR

- WriteData operation, when the corresponding Xxx_WriteData API is called in the Dcm Low Proxy,

  OR

- ReadDataLength operation, when the corresponding Xxx_ReadDataLength API is called in the Dcm Low Proxy,

  OR

- ConditionCheckRead operation, when the corresponding Xxx_ConditionCheckRead API is called in the Dcm Low Proxy,

  OR

- GetScalingInformation operation, when the corresponding Xxx_GetScalingInformation API is called in the Dcm Low Proxy,

  OR

- ReturnControlToECU operation, when the corresponding Xxx_ReturnControlToECU API is called in the Dcm Low Proxy,

  OR

- ResetToDefault operation, when the corresponding Xxx_ResetToDefault API is called in the Dcm Low Proxy,

  OR

- FreezeCurrentState operation, when the corresponding Xxx_FreezeCurrentState API is called in the Dcm Low Proxy,

  OR

- ShortTermAdjustment operation, when the corresponding Xxx_ShortTermAdjustment API is called in the Dcm Low Proxy.

In that case, IN arguments (Data (if applicable), OpStatus (if applicable), controlMask (if applicable) and ControlStateInfo (if applicable)) shall be passed to the client, and OUT arguments (Data (if applicable), ScalingInfo (if applicable), ErrorCode (if applicable)) and application errors shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02305]

Upstream requirements: SRS_SwCluC_00208

⌈In case the operations in the DataServices_{Data} ClientServerInterface do not have a controlMask argument configured, the value shall be discarded without checking the passed controlMaskLength.⌋

[SWS_SwCluC_02306]

Upstream requirements: SRS_SwCluC_00208

⌈In case the operations in the DataServices_{Data} ClientServerInterface do not have an ErrorCode argument configured, the default value DCM_POS_RESP shall be transferred back to the caller of the Dcm Low Proxy in case the caller returned E_OK.⌋

[SWS_SwCluC_02307]

Upstream requirements: SRS_SwCluC_00208

⌈In case the DataServices_{Data} RPortPrototype is unconnected, the application error E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02308]

Upstream requirements: SRS_SwCluC_00208

⌈In case the passed DataLength, ScalingInfoLength, or controlMaskLength (if applicable, see [SWS_SwCluC_02305]) of the caller of the Dcm Low Proxy do not match the configured data size in the Dcm configuration in the Application Software Cluster, the application error E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy. No data, scalingInfo or controlMask values shall be transferred in this case.⌋

Please note: In case the data has dynamic length configured in the Application Software Cluster for this diagnostic data element, the DataLength cannot be checked.

DataServices_{Data} via sender-receiver / nv-data interfaces

[SWS_SwCluC_02309]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{Data} RPortPrototypes for SenderReceiverInterface/NvDataInterface DataServices_{Data}.⌋

Note: In the corresponding Dcm configuration, DcmDspDataUsePort will usually be set to USE_DATA_SENDER_RECEIVER or USE_DATA_SENDER_RECEIVER_AS_SERVICE AND the related parameters DcmDspDidRead will not exist AND DcmDspDidWrite will not exist.

[SWS_SwCluC_02310]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{Data} PPortPrototypes for SenderReceiverInterface/NvDataInterface DataServices_{Data}.⌋

Note: In the corresponding Dcm configuration, DcmDspDataUsePort will usually be set to USE_DATA_SENDER_RECEIVER or USE_DATA_SENDER_RECEIVER_AS_SERVICE AND the related parameters DcmDspDidRead will not exist AND DcmDspDidWrite will exist.

[SWS_SwCluC_02311]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{Data} PRPortPrototypes for SenderReceiverInterface/NvDataInterface DataServices_{Data}.⌋

Note: In the corresponding Dcm configuration, DcmDspDataUsePort will usually be set to USE_DATA_SENDER_RECEIVER or USE_DATA_SENDER_RECEIVER_AS_SERVICE AND the related parameters DcmDspDidRead will exist AND DcmDspDidWrite will not exist.

[SWS_SwCluC_02312]

Upstream requirements: SRS_SwCluC_00208

⌈In case the SwCluCDcmProxyDiagnosticData channel related to a DataServices_{Data} PortPrototypes is connected, the Dcm Proxy shall

- read the dataElement/nvData, when the corresponding Xxx_ReadData API is called in the Dcm Low Proxy,

  OR

- write the dataElement/nvData, when the corresponding Xxx_WriteData API is called in the Dcm Low Proxy.

In that case IN argument Data (if applicable) shall be sent to the receiver, and OUT argument Data (if applicable) shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02313]

Upstream requirements: SRS_SwCluC_00208

⌈In case the DataServices_{Data} SenderReceiverInterface/NvDataInterface is configured, the default value DCM_POS_RESP as ErrorCode shall be transferred back to the caller of the Dcm Low Proxy for any connected RPortPrototype and any connected or unconnected PPortPrototype or PRPortPrototype.⌋

[SWS_SwCluC_02314]

Upstream requirements: SRS_SwCluC_00208

⌈In case the DataServices_{Data} RPortPrototype, PPortPrototype or PRPortPrototype is unconnected, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02315]

Upstream requirements: SRS_SwCluC_00208

⌈In case the passed DataLength of the caller of the Dcm Low Proxy does not match the configured data size in the Dcm configuration in the Application Software Cluster, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy. No data shall be transferred in this case.⌋

DataServices_{DID} ports

The DataServices_{DID} ports are used to access entire data belonging to a DID.

[SWS_SwCluC_02316]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{DID} RPortPrototypes for SenderReceiverInterface DataServices_{DID}.⌋

Note: In the corresponding Dcm configuration, DcmDspDidUsePort will usually be set to USE_ATOMIC_SENDER_RECEIVER_INTERFACE or USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE AND the related parameters DcmDspDidRead will exist AND DcmDspDidWrite will not exist OR DcmDspDidControl will exist.

[SWS_SwCluC_02317]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{DID} PPortPrototypes for SenderReceiverInterface DataServices_{DID}.⌋

Note: In the corresponding Dcm configuration, DcmDspDidUsePort will usually be set to USE_ATOMIC_SENDER_RECEIVER_INTERFACE or USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE AND the related parameters DcmDspDidRead will not exist AND DcmDspDidWrite will exist OR DcmDspDidControl will exist.

[SWS_SwCluC_02318]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of DataServices_{DID} PRPortPrototypes for SenderReceiverInterface DataServices_{DID}.⌋

Note: In the corresponding Dcm configuration, DcmDspDidUsePort will usually be set to USE_ATOMIC_SENDER_RECEIVER_INTERFACE, USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE, or USE_ATOMIC_NV_DATA_INTERFACE AND the related parameters DcmDspDidRead will exist AND DcmDspDidWrite will exist OR DcmDspDidControl will exist.

[SWS_SwCluC_02319]

Upstream requirements: SRS_SwCluC_00208

⌈In case the SwCluCDcmProxyDiagnosticData channel related to a DataServices_{DID} PortPrototypes is connected, the Dcm Proxy shall

- read the dataElement, when the corresponding Xxx_ReadData API is called in the Dcm Low Proxy,

  OR

- write the dataElement, when the corresponding Xxx_WriteData API is called in the Dcm Low Proxy,

In that case, IN argument Data (if applicable) shall be sent to the receiver and OUT argument Data (if applicable) shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02320]

Upstream requirements: SRS_SwCluC_00208

⌈In case the DataServices_{DID} SenderReceiverInterface is configured, the default value DCM_POS_RESP as ErrorCode shall be transferred back to the caller of the Dcm Low Proxy for any connected RPortPrototype and any connected or unconnected PPortPrototype or PRPortPrototype.⌋

[SWS_SwCluC_02321]

Upstream requirements: SRS_SwCluC_00208

⌈In case the DataServices_{DID} RPortPrototype is unconnected, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02322]

Upstream requirements: SRS_SwCluC_00208

⌈In case the passed DataLength of the caller of the Dcm Low Proxy does not match the configured data size in the Dcm configuration in the Application Software Cluster, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy. No data shall be transferred in this case.⌋

IOControlRequest_{DID} and IOControlResponse_{DID} ports

The IOControlRequest_{DID} and IOControlResponse_{DID} sender-receiver ports are used for the Service 0x2F - InputOutputControlByIdentifier as an alternative to the DataServices_{Data} client-server ports

[SWS_SwCluC_02323]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of IOControlRequest_{DID} RPortPrototypes for SenderReceiverInterface IOControlRequest_{DID}.⌋

[SWS_SwCluC_02324]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of IOControlResponse_{DID} RPortPrototypes for SenderReceiverInterface IOControlResponse_{DID}.⌋

Note: In the corresponding Dcm configuration, DcmDspDidUsePort will usually be set to USE_ATOMIC_SENDER_RECEIVER_INTERFACE or USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE and the related container DcmDspDid will exist.

[SWS_SwCluC_02325]

Upstream requirements: SRS_SwCluC_00208

⌈In case the SwCluCDcmProxyDiagnosticData channel, related to a IOControlRequest_{DID} and IOControlResponse_{DID} sender-receiver port, is connected, the Dcm Proxy shall:

- calculate the underControl, IOOperationRequest and controlState data elements, according to [SWS_Dcm_01435], [SWS_Dcm_00682], [SWS_Dcm_01436], [SWS_Dcm_01437], [SWS_Dcm_01438], [SWS_Dcm_01275], [SWS_Dcm_01277] AND
- return the IOOperationResponse, according to [SWS_Dcm_01439], [SWS_Dcm_01440], and [SWS_Dcm_01441], when the corresponding Xxx_ReadData, Xxx_WriteData, Xxx_ReturnControlToECU, Xxx_ResetToDefault, Xxx_FreezeCurrentState, or Xxx_ShortTermAdjustment API is called in the Dcm Low Proxy.

⌋

Binary Manifest

[SWS_SwCluC_02391]

Upstream requirements: SRS_SwCluC_00212

⌈The Dcm High Proxy shall create for each diagnostic data proxy - configured via SwCluCDcmProxyDiagnosticData container - a provided resource entry in the Binary Manifest.⌋

7.4.3.4.2.2 Routine Control proxy

The routine control proxy provides access to routine control services in Application Software Clusters, which can be accessed by UDS diagnostic services 0x31 RoutineControl via the Dcm in the Host Software Cluster.

RoutineServices_{RoutineName} ports

[SWS_SwCluC_02330]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of RoutineServices_{RoutineName} RPortPrototypes for ClientServerInterface RoutineServices_{RoutineName}.⌋

[SWS_SwCluC_02331]

Upstream requirements: SRS_SwCluC_00208

⌈In case the SwCluCDcmProxyRoutineControl channel related to a RoutineServices_{RoutineName} RPortPrototypes is connected, the Dcm Proxy shall invoke the

- Start operation, when the corresponding Xxx_Start API is called in the Dcm Low Proxy,

  OR

- StartConfirmation operation, when the corresponding Xxx_StartConfirmation API is called in the Dcm Low Proxy,

  OR

- Stop operation, when the corresponding Xxx_Stop API is called in the Dcm Low Proxy,

  OR

- StopConfirmation operation, when the corresponding Xxx_StopConfirmation API is called in the Dcm Low Proxy,

  OR

- RequestResults operation, when the corresponding Xxx_RequestResults API is called in the Dcm Low Proxy,

OR

- RequestResultsConfirmation operation, when the corresponding Xxx_RequestResultsConfirmation API is called in the Dcm Low Proxy,

In this case, INOUT arguments (dataInOut (if applicable)) are used to transfer the individual dataIn signals to the client, and to transfer back the individual dataOut signals as joint array to the caller of the Dcm Low Proxy. The configuration of the operations in the ClientServerInterfaces shall be according to the Dcm configuration of the Application Software Cluster. Additionally, IN arguments OpStatus (if applicable) and ConfirmationStatus (if applicable) shall be passed to the client, and OUT arguments ErrorCode (if applicable) and application errors shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02332]

Upstream requirements: SRS_SwCluC_00208

⌈In case the RoutineServices_{RoutineName} RPortPrototype is unconnected, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE (if applicable) shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02333]

Upstream requirements: SRS_SwCluC_00208

⌈In case the passed DataLength of the caller of the Dcm Low Proxy does not match the configured data size in the Dcm configuration in the Application Software Cluster, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE shall be transferred back to the caller of the Dcm Low Proxy. No data shall be transferred in this case.⌋

Please note: In case dynamic length signals are configured in the Application Software Cluster for this routine control, the DataLength cannot be checked.

Binary Manifest

[SWS_SwCluC_02392]

Upstream requirements: SRS_SwCluC_00212

⌈The Dcm High Proxy shall create for each routine control proxy - configured via SwCluCDcmProxyRoutineControl container - a provided resource entry in the Binary Manifest.⌋

7.4.3.4.2.3 Service Request Notification proxy

The service request notification proxy provides access to the service request manufacturer notification and service request supplier notification

ServiceRequestManufacturerNotification_{Name} ports

[SWS_SwCluC_02334]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00201

⌈The Dcm High Proxy shall support the configuration of ServiceRequestManufacturerNotification_{Name} RPortPrototypes for ClientServerInterface ServiceRequestManufacturerNotification_{Name}.⌋

ServiceRequestSupplierNotification_{Name} ports

[SWS_SwCluC_02335]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm High Proxy shall support the configuration of ServiceRequestSupplierNotification_{Name} RPortPrototypes for ClientServerInterface ServiceRequestSupplierNotification_{Name}.⌋

[SWS_SwCluC_02336]

Upstream requirements: SRS_SwCluC_00208

⌈In case the SwCluCDcmProxyServiceRequestNotification channel related to a ServiceRequestManufacturerNotification_{Name} or ServiceRequestSupplierNotification_{Name} RPortPrototypes is connected, the Dcm Proxy shall invoke the

- Confirmation operation, when the corresponding Xxx_Confirmation API is called in the Dcm Low Proxy,

  OR

- Indication operation, when the corresponding Xxx_Indication API is called in the Dcm Low Proxy.

In this case, IN arguments [SID (if applicable), ReqType (if applicable), ConnectionId (if applicable), ConfirmationStatus (if applicable), ProtocolType (if applicable), TesterSourceAddress (if applicable), RequestData (if applicable), DataSize (if applicable)] shall be passed to the client and OUT arguments [ErrorCode (if applicable)] and application errors shall be transferred back to the caller of the Dcm Low Proxy.⌋

[SWS_SwCluC_02337]

Upstream requirements: SRS_SwCluC_00208

⌈In case the ServiceRequestManufacturerNotification_{Name} or ServiceRequestSupplierNotification_{Name} RPortPrototype is unconnected, the return value E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE (if applicable) shall be transferred back to the caller of the Dcm Low Proxy.⌋

Binary Manifest

[SWS_SwCluC_02393]

Upstream requirements: SRS_SwCluC_00212

⌈The Dcm High Proxy shall create for each service request notification proxy - configured via SwCluCDcmProxyServiceRequestNotification container - a provided resource entry in the Binary Manifest.⌋

7.4.3.4.3 Dcm Low Proxy

7.4.3.4.3.1 Diagnostic Data proxy

ReadData

[SWS_SwCluC_02350]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_ReadData function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element read access is enabled.⌋

[SWS_SwCluC_02351]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_ReadData function shall be named SwCluC_DcmProxy_ReadData_<DDSN>, where <DDSN> is the shortName of the SwCluCDcmProxyDiagnosticData container.⌋

[SWS_SwCluC_02352]

Upstream requirements: SRS_SwCluC_00208

⌈The diagnostic data element read access for a SwCluCDcmProxyDiagnosticData shall be enabled, if:

- SwCluCDcmProxyDataRef references a DcmDspData AND the corresponding DcmDspDid has a DcmDspDidRead configured in its related DcmDspDidInfo

  OR

- SwCluCDcmProxyDataRef references a DcmDspDid, which has a DcmDspDidRead configured in its related DcmDspDidInfo

  OR

- SwCluCDcmProxyDataRef references a DcmDspPidService01

  OR

- SwCluCDcmProxyDataRef references a DemDataElementClass

⌋

[SWS_SwCluC_02380]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_ReadData function of Dcm Low Proxy shall return E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE, if the corresponding diagnostic data proxy is not connected to an instance in the Dcm High Proxy. No data shall be returned in this case.⌋

WriteData

[SWS_SwCluC_02353]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_WriteData function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element write access is enabled.⌋

[SWS_SwCluC_02354]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_WriteData function shall be named SwCluC_DcmProxy_WriteData_<DDSN>, where <DDSN> is the shortName of the SwCluCDcmProxyDiagnosticData container.⌋

[SWS_SwCluC_02355]

Upstream requirements: SRS_SwCluC_00208

⌈The diagnostic data element write access for a SwCluCDcmProxyDiagnosticData shall be enabled, if:

- SwCluCDcmProxyDataRef references a DcmDspData AND the corresponding DcmDspDid has a DcmDspDidWrite configured in its related DcmDspDidInfo

  OR

- SwCluCDcmProxyDataRef references a DcmDspDid, which has a DcmDspDidWrite configured in its related DcmDspDidInfo

⌋

[SWS_SwCluC_02381]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_WriteData function of Dcm Low Proxy shall return E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE, if the corresponding diagnostic data proxy is not connected to an instance in the Dcm High Proxy.⌋

ReadDataLength

[SWS_SwCluC_02356]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_ReadDataLength function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element read data length access is enabled.⌋

[SWS_SwCluC_02357]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_ReadDataLength function shall be named SwCluC_DcmProxy_ReadDataLength_<DDSN>, where <DDSN> is the shortName of the SwCluCDcmProxyDiagnosticData container.⌋

[SWS_SwCluC_02358]

Upstream requirements: SRS_SwCluC_00208

⌈The diagnostic data element read data length access for a SwCluCDcmProxyDiagnosticData shall be enabled, if:

- SwCluCDcmProxyDataRef references a DcmDspData AND the corresponding DcmDspDid has a DcmDspDidRead configured in its related DcmDspDidInfo

  AND

- the DcmDspDataType is set to UINT8_DYN.

⌋

[SWS_SwCluC_02382]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_ReadDataLength function of Dcm Low Proxy shall return E_NOT_OK, if the corresponding diagnostic data proxy is not connected to an instance in the Dcm High Proxy.⌋

ConditionCheckRead

[SWS_SwCluC_02359]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_ConditionCheckRead function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element condition check read access is enabled.⌋

[SWS_SwCluC_02360]

Upstream requirements: SRS_SwCluC_00208

⌈The `Xxx_ConditionCheckRead` function shall be named `SwCluC_DcmProxy_ConditionCheckRead_<DDSN>`, where `<DDSN>` is the `shortName` of the `SwCluCDcmProxyDiagnosticData` container.⌋

[SWS_SwCluC_02361]

Upstream requirements: SRS_SwCluC_00208

⌈The `diagnostic data element condition check read access` for a `SwCluCDcmProxyDiagnosticData` shall be enabled, if:

- `SwCluCDcmProxyDataRef` references a `DcmDspData` AND the corresponding `DcmDspDid` has a `DcmDspDidRead` configured in its related `DcmDspDidInfo`

  AND

- the parameter `DcmDspDataConditionCheckReadFncUsed` is set to `true`.

⌋

[SWS_SwCluC_02383]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The `Xxx_ConditionCheckRead` function of `Dcm Low Proxy` shall return `E_NOT_OK`, if the corresponding `diagnostic data proxy` is not connected to an instance in the `Dcm High Proxy`.⌋

GetScalingInformation

[SWS_SwCluC_02362]

Upstream requirements: SRS_SwCluC_00208

⌈The `Dcm Low Proxy` shall provide an instance of a `Xxx_GetScaling-Information` function per configured `SwCluCDcmProxyDiagnosticData` container in the `Dcm Low Proxy`, if the `diagnostic data element scaling information access` is enabled.⌋

[SWS_SwCluC_02363]

Upstream requirements: SRS_SwCluC_00208

⌈The `Xxx_GetScalingInformation` function shall be named `SwCluC_DcmProxy_GetScalingInformation_<DDSN>`, where `<DDSN>` is the `shortName` of the `SwCluCDcmProxyDiagnosticData` container.⌋

[SWS_SwCluC_02364]

Upstream requirements: SRS_SwCluC_00208

⌈The `diagnostic data element scaling information access` for a `SwCluCDcmProxyDiagnosticData` shall be enabled, if:

- SwCluCDcmProxyDataRef references a DcmDspData

  AND

- the corresponding DcmDspDataScalingInfoSize is configured in its related DcmDspDataInfo.

⌋

[SWS_SwCluC_02384]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_GetScalingInformation function of Dcm Low Proxy shall return E_NOT_OK, if the corresponding diagnostic data proxy is not connected to an instance in the Dcm High Proxy.⌋

ReturnControlToECU

[SWS_SwCluC_02365]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_ReturnControlToECU function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element return control to ECU is enabled.⌋

[SWS_SwCluC_02366]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_ReturnControlToECU function shall be named SwCluC_DcmProxy_ReturnControlToECU_<DDSN>, where <DDSN> is the shortName of the SwCluCDcmProxyDiagnosticData container.⌋

[SWS_SwCluC_02367]

Upstream requirements: SRS_SwCluC_00208

⌈The diagnostic data element return control to ECU for a SwCluCDcmProxyDiagnosticData shall be enabled, if:

- SwCluCDcmProxyDataRef references a DcmDspData

  AND

- the corresponding DcmDspDid has a DcmDspDidControl configured in its related DcmDspDidInfo

  AND

- the parameter DcmDspDidFreezeCurrentState is set to true.

⌋

[SWS_SwCluC_02385]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The `Xxx_ReturnControlToECU` function of `Dcm Low Proxy` shall return `E_NOT_OK`, if the corresponding `diagnostic data proxy` is not connected to an instance in the `Dcm High Proxy`.⌋

ResetToDefault

[SWS_SwCluC_02368]

Upstream requirements: SRS_SwCluC_00208

⌈The `Dcm Low Proxy` shall provide an instance of a `Xxx_ResetToDefault` function per configured `SwCluCDcmProxyDiagnosticData` container in the `Dcm Low Proxy`, if the `diagnostic data element reset to default` is enabled.⌋

[SWS_SwCluC_02369]

Upstream requirements: SRS_SwCluC_00208

⌈The `Xxx_ResetToDefault` function shall be named `SwCluC_DcmProxy_ResetToDefault_<DDSN>`, where `<DDSN>` is the `shortName` of the `SwCluCDcmProxyDiagnosticData` container.⌋

[SWS_SwCluC_02370]

Upstream requirements: SRS_SwCluC_00208

⌈The `diagnostic data element reset to default` for a `SwCluCDcmProxyDiagnosticData` shall be enabled, if:

- `SwCluCDcmProxyDataRef` references a `DcmDspData`

  AND

- the corresponding `DcmDspDid` has a `DcmDspDidControl` configured in its related `DcmDspDidInfo`

  AND

- the parameter `DcmDspDidResetToDefault` is set to `true`.

⌋

[SWS_SwCluC_02386]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The `Xxx_ResetToDefault` function of `Dcm Low Proxy` shall return `E_NOT_OK`, if the corresponding `diagnostic data proxy` is not connected to an instance in the `Dcm High Proxy`.⌋

FreezeCurrentState

[SWS_SwCluC_02371]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_FreezeCurrentState function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element freeze current state is enabled.⌋

[SWS_SwCluC_02372]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_FreezeCurrentState function shall be named SwCluC_DcmProxy_FreezeCurrentState_<DDSN>, where <DDSN> is the shortName of the SwCluCDcmProxyDiagnosticData container.⌋

[SWS_SwCluC_02373]

Upstream requirements: SRS_SwCluC_00208

⌈The diagnostic data element freeze current state for a SwCluCDcmProxyDiagnosticData shall be enabled, if

- SwCluCDcmProxyDataRef references a DcmDspData

  AND

- the corresponding DcmDspDid has a DcmDspDidControl configured in its related DcmDspDidInfo

  AND

- the parameter DcmDspDidFreezeCurrentState is set to true.

⌋

[SWS_SwCluC_02387]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_FreezeCurrentState function of Dcm Low Proxy shall return E_NOT_OK, if the corresponding diagnostic data proxy is not connected to an instance in the Dcm High Proxy.⌋

ShortTermAdjustment

[SWS_SwCluC_02374]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_ShortTermAdjustment function per configured SwCluCDcmProxyDiagnosticData container in the Dcm Low Proxy, if the diagnostic data element short term adjustment is enabled.⌋

[SWS_SwCluC_02375]

Upstream requirements: SRS_SwCluC_00208

⌈The `Xxx_ShortTermAdjustment` function shall be named `SwCluC_DcmProxy_ShortTermAdjustment_<DDSN>`, where `<DDSN>` is the `shortName` of the `SwCluCDcmProxyDiagnosticData` container.⌋

[SWS_SwCluC_02376]

Upstream requirements: SRS_SwCluC_00208

⌈The `diagnostic data element short term adjustment` for a `SwCluCDcmProxyDiagnosticData` shall be enabled, if

- `SwCluCDcmProxyDataRef` references a `DcmDspData`

  AND

- the corresponding `DcmDspDid` has a `DcmDspDidControl` configured in its related `DcmDspDidInfo`

  AND

- the parameter `DcmDspDidShortTermAdjustment` is set to `true`.

⌋

[SWS_SwCluC_02388]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The `Xxx_ShortTermAdjustment` function of `Dcm Low Proxy` shall return `E_NOT_OK`, if the corresponding `diagnostic data proxy` is not connected to an instance in the `Dcm High Proxy`.⌋

Binary Manifest

[SWS_SwCluC_02394]

Upstream requirements: SRS_SwCluC_00212

⌈The `Dcm Low Proxy` shall create for each `diagnostic data proxy` - configured via `SwCluCDcmProxyDiagnosticData` container - a required resource entry in the `Binary Manifest`.⌋

7.4.3.4.3.2 Routine Control proxy

Start Operation

[SWS_SwCluC_02400]

Upstream requirements: SRS_SwCluC_00208

⌈The `Dcm Low Proxy` shall provide an instance of a `Xxx_Start` function per configured `SwCluCDcmProxyRoutineControl` container in the `Dcm Low Proxy`.⌋

[SWS_SwCluC_02401]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_Start function shall be named SwCluC_DcmProxy_Start_<RCSN>, where <RCSN> is the shortName of the SwCluCDcmProxyRoutineControl container.⌋

[SWS_SwCluC_02402]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_Start function of Dcm Low Proxy shall return E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE, if the corresponding routine control proxy is not connected to an instance in the Dcm High Proxy.⌋

StartConfirmation Operation

[SWS_SwCluC_02403]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_StartConfirmation function per configured SwCluCDcmProxyRoutineControl container in the Dcm Low Proxy.⌋

[SWS_SwCluC_02404]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_StartConfirmation function shall be named SwCluC_DcmProxy_StartConfirmation_<RCSN>, where <RCSN> is the shortName of the SwCluCDcmProxyRoutineControl container.⌋

[SWS_SwCluC_02405]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_StartConfirmation function of Dcm Low Proxy shall return E_NOT_OK, if the corresponding routine control proxy is not connected to an instance in the Dcm High Proxy.⌋

Stop Operation

[SWS_SwCluC_02406]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_Stop function per configured SwCluCDcmProxyRoutineControl container in the Dcm Low Proxy.⌋

[SWS_SwCluC_02407]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_Stop function shall be named SwCluC_DcmProxy_Stop_<RCSN>, where <RCSN> is the shortName of the SwCluCDcmProxyRoutineControl container.⌋

[SWS_SwCluC_02408]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_Stop function of Dcm Low Proxy shall return E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE, if the corresponding routine control proxy is not connected to an instance in the Dcm High Proxy.⌋

StopConfirmation Operation

[SWS_SwCluC_02409]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_StopConfirmation function per configured SwCluCDcmProxyRoutineControl container in the Dcm Low Proxy.⌋

[SWS_SwCluC_02410]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_StopConfirmation function shall be named SwCluC_DcmProxy_StopConfirmation_<RCSN>, where <RCSN> is the shortName of the SwCluCDcmProxyRoutineControl container.⌋

[SWS_SwCluC_02411]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_StopConfirmation function of Dcm Low Proxy shall return E_NOT_OK if the corresponding routine control proxy is not connected to an instance in the Dcm High Proxy.⌋

RequestResults Operation

[SWS_SwCluC_02412]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_RequestResults function per configured SwCluCDcmProxyRoutineControl container in the Dcm Low Proxy.⌋

[SWS_SwCluC_02413]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_RequestResults function shall be named SwCluC_DcmProxy_RequestResults_<RCSN>, where <RCSN> is the shortName of the SwCluCDcmProxyRoutineControl container.⌋

[SWS_SwCluC_02414]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_RequestResults function of Dcm Low Proxy shall return E_NOT_OK and the ErrorCode DCM_E_REQUESTOUTOFRANGE, if the corresponding routine control proxy is not connected to an instance in the Dcm High Proxy.⌋

RequestResultsConfirmation Operation

[SWS_SwCluC_02415]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_RequestResults-Confirmation function per configured SwCluCDcmProxyRoutineControl container in the Dcm Low Proxy.⌋

[SWS_SwCluC_02416]

Upstream requirements: SRS_SwCluC_00208

⌈The Xxx_RequestResultsConfirmation function shall be named SwCluC_DcmProxy_RequestResultsConfirmation_<RCSN>, where <RCSN> is the shortName of the SwCluCDcmProxyRoutineControl container.⌋

[SWS_SwCluC_02417]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The Xxx_RequestResultsConfirmation function of Dcm Low Proxy shall return E_NOT_OK, if the corresponding routine control proxy is not connected to an instance in the Dcm High Proxy.⌋

Binary Manifest

[SWS_SwCluC_02395]

Upstream requirements: SRS_SwCluC_00212

⌈The Dcm Low Proxy shall create for each routine control proxy - configured via SwCluCDcmProxyRoutineControl container - a required resource entry in the Binary Manifest.⌋

7.4.3.4.3.3 Service Request Notification proxy

Indication Operation

[SWS_SwCluC_02418]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide an instance of a Xxx_Indication function per configured SwCluCDcmProxyServiceRequestNotification container in the Dcm Low Proxy.⌋

[SWS_SwCluC_02419]

Upstream requirements: SRS_SwCluC_00208

⌈The `Xxx_Indication` function shall be named `SwCluC_DcmProxy_Indication_<SRNSN>`, where `<SRNSN>` is the `shortName` of the `SwCluCDcmProxyServiceRequestNotification` container.⌋

[SWS_SwCluC_02420]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The `Xxx_Indication` function of `Dcm Low Proxy` shall return `E_NOT_OK` and the `ErrorCode` `DCM_E_REQUESTOUTOFRANGE`, if the corresponding `routine control proxy` is not connected to an instance in the `Dcm High Proxy`.⌋

Confirmation Operation

[SWS_SwCluC_02421]

Upstream requirements: SRS_SwCluC_00208

⌈The `Dcm Low Proxy` shall provide an instance of a `Xxx_Confirmation` function per configured `SwCluCDcmProxyServiceRequestNotification` container in the `Dcm Low Proxy`.⌋

[SWS_SwCluC_02422]

Upstream requirements: SRS_SwCluC_00208

⌈The `Xxx_Confirmation` function shall be named `SwCluC_DcmProxy_Confirmation_<SRNSN>`, where `<SRNSN>` is the `shortName` of the `SwCluCDcmProxyServiceRequestNotification` container.⌋

[SWS_SwCluC_02423]

Upstream requirements: SRS_SwCluC_00208, SRS_SwCluC_00213

⌈The `Xxx_Confirmation` function of `Dcm Low Proxy` shall return `E_NOT_OK` and the `ErrorCode` `DCM_E_REQUESTOUTOFRANGE`, if the corresponding `routine control proxy` is not connected to an instance in the `Dcm High Proxy`.⌋

Binary Manifest

[SWS_SwCluC_02396]

Upstream requirements: SRS_SwCluC_00212

⌈The `Dcm Low Proxy` shall create for each `service request notification proxy` - configured via `SwCluCDcmProxyServiceRequestNotification` container - a required resource entry in the `Binary Manifest`.⌋

7.4.3.4.3.4 Multiple Partitions

The invoked interface functions need to change the partition in the Host Software Cluster, before the calls are ending up in the Application Software Cluster, if the Dcm module does not offer a satellite on this partition. This partition change can be implemented in the Low Proxy, by means of the SchM or with platform proprietary callouts.

[SWS_SwCluC_02425]

Upstream requirements: SRS_SwCluC_00208

⌈The Dcm Low Proxy shall provide, for each SwCluCDcmProxyBaseSocket, a SwCluC_DcmProxy_Init API, to initialize the Dcm High Proxy of an Application Software Cluster. This shall be one element of the service resource SwCluCDcmProxyBaseSocket.⌋

[SWS_SwCluC_02424]

Upstream requirements: SRS_SwCluC_00208

⌈The interface function instance for the diagnostic data proxy, routine control proxy and service request notification proxy, according to [SWS_SwCluC_02350], [SWS_SwCluC_02353], [SWS_SwCluC_02356], [SWS_SwCluC_02359], [SWS_SwCluC_02362], [SWS_SwCluC_02365], [SWS_SwCluC_02368], [SWS_SwCluC_02371], [SWS_SwCluC_02374], [SWS_SwCluC_02400], [SWS_SwCluC_02403], [SWS_SwCluC_02406], [SWS_SwCluC_02409], [SWS_SwCluC_02412], [SWS_SwCluC_02415], shall be provided either:

- at the EcucPartition, referenced by
  - SwCluCDcmProxyDiagnosticData.SwCluCProxyEcucPartitionRef

    OR
  - SwCluCDcmProxyRoutineControl.SwCluCProxyEcucPartitionRef

    OR
  - SwCluCDcmProxyServiceRequestNotification.SwCluCProxyEcucPartitionRef

  OR
- if a SwCluCDcmProxyBaseSocket is directly available, at its EcucPartition

  OR
- at the EcucPartition of the SwCluCDcmProxyBaseSocket, referenced by the SwCluCDcmProxyBaseSocket - via SwCluCDcmProxyUsedSatelliteRef, applicable for the EcucPartition requested above. In this case, the EcucPartition shall be changed, before the callback in the High Proxy is called.

⌋

Binary Manifest

The implementation of the Dcm Low Proxy may provide some basic infrastructure for the Dcm High Proxy in order to be operational. Such generic infrastructure services can be linked between Application Software Cluster and Host Software Cluster by means of a Dcm base socket.

[SWS_SwCluC_02397]

Upstream requirements: SRS_SwCluC_00212

⌈The Dcm Low Proxy shall create for each SwCluCDcmProxyBaseSocket a provided resource entry in the Binary Manifest.⌋

7.4.3.5 Dem Proxy

The specified Dem Proxy has the following underlying design principle:

In brief, the core business of Dem is the management of Diagnostic Events and the event memory, used to store failures and to capture related freeze frames with environmental data. This requires mainly interfaces to report and query status Diagnostic Events, as well as interfaces to read environmental data for the freeze frames. The Dem Proxy provides the ability implement so called diagnostic event proxies, enabling that Diagnostic Monitors - reporting diagnostic events - in the Application Software Clusters can access the Dem in the Host Software Cluster. APIs and AUTOSAR Service Interfaces that are not related to Diagnostic Monitors may not be supported by the Dem Proxy. Data relevant for freeze frames are (according diagnostic standards) are also accessible via Dcm by DID. Hence, the diagnostic data proxy functionality of the Dcm Proxy has to be used.

7.4.3.5.1 Enable Dem Proxy Generation

[SWS_SwCluC_02748]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00204

⌈The Dem High Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationDem is set to HIGH_PROXY.⌋

[SWS_SwCluC_02749]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00204

⌈The Dem Low Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationDem is set to LOW_PROXY.⌋

7.4.3.5.2 Dem High Proxy

7.4.3.5.2.1 Dem High Proxy C-API

[SWS_SwCluC_02700]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem High Proxy shall provide the header file Dem.h.⌋

[SWS_SwCluC_02701]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem High Proxy shall provide the symbolic name values of DemEventIds for Diagnostic Monitors in the Application Software Cluster.⌋

[SWS_SwCluC_02702]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall provide the functions for Diagnostic Monitors reporting Diagnostic Events

- Dem_ClearPrestoredFreezeFrame
- Dem_PrestoreFreezeFrame
- Dem_ResetEventDebounceStatus
- Dem_ResetEventStatus
- Dem_SetEventDisabled
- Dem_SetEventStatus

in the Application Software Cluster.⌋

[SWS_SwCluC_02736]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall provide the function Dem_SetEventStatusWithMonitorData, for Diagnostic Monitors reporting Diagnostic Events with monitor data in the Application Software Cluster.⌋

[SWS_SwCluC_02739]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall provide the function Dem_SetWIRStatus, for Diagnostic Monitors reporting Diagnostic Events in the Application Software Cluster.⌋

[SWS_SwCluC_02703]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201, SRS_SwCluC_00202

⌈The Dem High Proxy shall provide the functions for Diagnostic Info of Diagnostic Events

- Dem_GetDTCOfEvent
- Dem_GetDebouncingOfEvent
- Dem_GetEventAvailable
- Dem_GetEventExtendedDataRecordEx
- Dem_GetEventFreezeFrameDataEx
- Dem_GetEventUdsStatus
- Dem_GetFaultDetectionCounter
- Dem_GetMonitorStatus

in the Application Software Cluster.⌋

[SWS_SwCluC_02751]

Upstream requirements: SRS_SwCluC_00209, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If development error detection is enabled for Dem High Proxy, the APIs

- Dem_ClearPrestoredFreezeFrame
- Dem_PrestoreFreezeFrame
- Dem_ResetEventDebounceStatus
- Dem_ResetEventStatus
- Dem_SetEventDisabled
- Dem_SetEventStatus
- Dem_GetDTCOfEvent
- Dem_GetDebouncingOfEvent
- Dem_GetEventAvailable
- Dem_GetEventExtendedDataRecordEx
- Dem_GetEventFreezeFrameDataEx
- Dem_GetEventUdsStatus
- Dem_GetFaultDetectionCounter
- Dem_GetMonitorStatus

- Dem_SetEventStatusWithMonitorData
- Dem_SetWIRStatus

shall report the DET error DEM_E_UNINIT, when Dem High Proxy is not yet initialized.⌋

7.4.3.5.2.2 Callback Functions

[SWS_SwCluC_02704]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall support the optional configuration of a <Module>_DemGetFaultDetectionCounter<ForEvent> callback, according to [ECUC_Dem_00630] and [ECUC_Dem_00631].⌋

[SWS_SwCluC_02705]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall support the optional configuration of a <Module>_DemInitMonitorFor<EventName> callback, according to [ECUC_Dem_00632] and [ECUC_Dem_00601].⌋

[SWS_SwCluC_02706]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall support the optional configuration of a <Module>_DemTriggerOnMonitorStatus callback, according to [ECUC_Dem_00936] and [ECUC_Dem_00937].⌋

[SWS_SwCluC_02707]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall support the optional configuration of a <Module>_DemClearEventAllowed<ForCondition> callback, according to [ECUC_Dem_00607] and [ECUC_Dem_00609].⌋

[SWS_SwCluC_02737]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall support the optional configuration of a <Module>_DemTriggerOnEventData callback, according to [ECUC_Dem_00606] and [ECUC_Dem_00608].⌋

[SWS_SwCluC_02738]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00202

⌈The Dem High Proxy shall support the optional configuration of a <Module>_DemTriggerOnEventUdsStatus callback, according to [ECUC_Dem_00628] and [ECUC_Dem_00629].⌋

7.4.3.5.2.3 Service Ports

[SWS_SwCluC_02723]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the ports belonging to a specific Diagnostic Event

- Event_{Name}
- EventStatus_{Name}

at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02724]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the DiagnosticMonitor_MonitorData_{Name} port for each specific Diagnostic Event, where DemEventReportingType is set to STANDARD_REPORTING_WITH_MONITOR_DATA at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02740]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the EventStatus_{Name} port for each specific Diagnostic Event at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02725]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the CBClrEvt_{Name} port for each specific Diagnostic Event, where DemCallbackClearEventAllowed is configured at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02726]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the CBFaultDetectCtr_{Name} port for each specific Diagnostic Event, where DemCallbackGetFDC is configured at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

Note: this applies only if DemDebounceAlgorithmClass is set to DebounceMonitorInternal

[SWS_SwCluC_02727]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the CBInitEvt_{Name} port for each specific Diagnostic Event, where DemCallbackInitMForE is configured at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02741]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the CBDataEvt_{Name} port for each specific Diagnostic Event, where DemCallbackEventDataChanged is configured at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02742]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the CBEventUdsStatusChanged_{EventName}_{CallbackName} port for each specific Diagnostic Event, where DemCallbackEventUdsStatusChanged is configured at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

[SWS_SwCluC_02743]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00201

⌈The Dem High Proxy shall provide the CBMonitorStatusChanged_{EventName}_{CallbackName} port for each specific Diagnostic Event, where DemCallbackMonitorStatusChanged is configured at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

Multiple Partitions

[SWS_SwCluC_02708]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem High Proxy shall create one partition specific Service Software Component per configured EcucPartition, with the name Dem_<EcucPartition shortName>.⌋

[SWS_SwCluC_02709]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem High Proxy shall provide the ports belonging to a specific interface connection at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋

7.4.3.5.2.4 Binary Manifest

The implementation of the Dem High Proxy may require some basic infrastructure from the Dem Low Proxy in order to be operational. Such generic infrastructure services can be linked between Application Software Cluster and Host Software Cluster by means of a Dem base socket.

[SWS_SwCluC_02744]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00212

⌈The Dem High Proxy shall create for each SwCluCDemProxyBaseSocket a required resource entry in the Binary Manifest.⌋

[SWS_SwCluC_02745]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00212

⌈The Dem High Proxy shall create for each diagnostic event proxy - configured via SwCluCDemProxyDiagnosticEvent container - a required resource entry in the Binary Manifest.⌋

7.4.3.5.2.5 Handling of unconnected Diagnostic Events

[SWS_SwCluC_02710]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the functions / operations

- Dem_ClearPrestoredFreezeFrame / ClearPrestoredFreezeFrame
- Dem_PrestoreFreezeFrame / PrestoreFreezeFrame
- Dem_ResetEventDebounceStatus / ResetEventDebounceStatus

- Dem_ResetEventStatus / ResetEventStatus
- Dem_SetEventDisabled / SetEventDisabled
- Dem_SetEventStatus / SetEventStatus
- Dem_SetEventStatusWithMonitorData / SetEventStatusWithMonitorData
- Dem_SetWIRStatus / SetWIRStatus

of the Dem High Proxy shall return E_NOT_OK.⌋

[SWS_SwCluC_02711]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return DEM_E_NO_DTC_AVAILABLE and the value 0 for the DTCOfEvent argument, when Dem_GetDTCOfEvent / GetDTCOfEvent is called.⌋

[SWS_SwCluC_02712]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value 0 for the DebouncingState argument, when Dem_GetDebouncingOfEvent / GetDebouncingOfEvent is called.⌋

[SWS_SwCluC_02713]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value FALSE for the AvailableStatus argument, when Dem_GetEventAvailable / GetEventAvailable is called.⌋

[SWS_SwCluC_02714]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value 0 for the Bufsize argument, when Dem_GetEventExtendedDataRecordEx / GetEventExtendedDataRecordEx is called. The Bufsize argument stays unmodified.⌋

[SWS_SwCluC_02715]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value

0 for the Bufsize argument, when Dem_GetEventFreezeFrameDataEx / GetEventFreezeFrameDataEx is called. The Bufsize argument stays unmodified.⌋

[SWS_SwCluC_02716]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value 0 for the UDSStatusByte argument, when Dem_GetEventUdsStatus / GetEventUdsStatus is called.⌋

[SWS_SwCluC_02717]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value 0 for the FaultDetectionCounter argument, when Dem_GetFaultDetectionCounter / GetFaultDetectionCounter is called.⌋

[SWS_SwCluC_02718]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem High Proxy is not connected to any Dem Low Proxy, the Dem High Proxy shall return E_NOT_OK and the value 0 for the MonitorStatus argument, when Dem_GetMonitorStatus / GetMonitorStatus is called.⌋

7.4.3.5.3 Dem Low Proxy

As shown in figure 7.10, the Dem Low Proxy shall be able to invoke APIs of the BSW modules in the Host Software Cluster, without a partition change in the Application Software Clusters. This partition change is only needed, if the BSW module does not offer a satellite on this partition. The availability of the BSW modules on dedicated EcucPartitions is a property of the Host Software Cluster.

[SWS_SwCluC_02719]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy shall provide the entry functions for Diagnostic Event reporting and info, according to [SWS_SwCluC_02702] and [SWS_SwCluC_02703], once per configured SwCluCDemProxyBaseSocket at the EcucPartition configured by SwCluCDemProxyBaseSocket.SwCluCProxyEcucPartitionRef.
In case the SwCluCDemProxyUsedSatelliteRef is set, the EcucPartition shall be changed, before the Dem API is called.⌋

[SWS_SwCluC_02728]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy shall provide, for each SwCluCDemProxyBaseSocket, a SwCluC_DemProxy_Init API, to initialize the Dem High Proxy of an Application Software Cluster. This shall be one element of the service resource SwCluCDemProxyBaseSocket.⌋

[SWS_SwCluC_02729]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy shall provide an instance of a <Module>_DemGetFaultDetectionCounter<ForEvent> per configured Diagnostic Event (SwCluCDemProxyDiagnosticEvent), in the Dem Low Proxy, if DemDebounceAlgorithmClass is set to DebounceMonitorInternal.⌋

[SWS_SwCluC_02730]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy shall provide an instance of a <Module>_DemInitMonitorFor<EventName> per configured Diagnostic Event (SwCluCDemProxyDiagnosticEvent), in the Dem Low Proxy.⌋

[SWS_SwCluC_02731]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy shall provide an instance of a <Module>_DemTriggerOnMonitorStatus per configured Diagnostic Event (SwCluCDemProxyDiagnosticEvent), in the Dem Low Proxy.⌋

[SWS_SwCluC_02732]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy shall provide an instance of a <Module>_DemClearEventAllowed<ForCondition> per configured Diagnostic Event (SwCluCDemProxyDiagnosticEvent), in the Dem Low Proxy.⌋

[SWS_SwCluC_02733]

Upstream requirements: SRS_SwCluC_00209

⌈The Dem Low Proxy implementation shall describe the Diagnostic Eventss provided to Dem High Proxys via DiagnosticEventNeeds and DiagnosticEventInfoNeeds, derived from the CpSoftwareClusterServiceResource.resourceNeeds.⌋

Note: [SWS_SwCluC_02733] ensures a partly automated configuration of the Diagnostic Event Manager.

7.4.3.5.3.1 Binary Manifest

The implementation of the Dem Low Proxy may provide some basic infrastructure for the Dem High Proxy in order to be operational. Such generic infrastructure services can be linked between Application Software Cluster and Host Software Cluster by means of a Dem base socket.

[SWS_SwCluC_02746]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00212

⌈The Dem Low Proxy shall create, for each SwCluCDemProxyBaseSocket, a provided resource entry in the Binary Manifest.⌋

[SWS_SwCluC_02747]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00212

⌈The Dem Low Proxy shall create for, each diagnostic event proxy - configured via SwCluCDemProxyDiagnosticEvent container -, a provided resource entry in the Binary Manifest.⌋

7.4.3.5.3.2 Handling of unconnected Diagnostic Events

[SWS_SwCluC_02734]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem Low Proxy is not connected to any Dem High Proxy, the callback functions

- <Module>_DemGetFaultDetectionCounter<ForEvent>
- <Module>_DemClearEventAllowed<ForCondition>

of the Dem High Proxy shall return E_NOT_OK.⌋

Note: The callback functions in [SWS_SwCluC_02734] indicate that the call to the diagnostic monitor has failed, in case the Diagnostic Event is not connected.

[SWS_SwCluC_02735]

Upstream requirements: SRS_SwCluC_00209, SRS_SwCluC_00213

⌈In case a Diagnostic Event configured in the Dem Low Proxy is not connected to any Dem High Proxy, the callback functions

- <Module>_DemInitMonitorFor<EventName>
- <Module>_DemTriggerOnMonitorStatus

of the Dem High Proxy shall return E_OK.⌋

Note: The callback functions in [SWS_SwCluC_02735] do not support an error code different than E_OK.

7.4.3.6 FiM Proxy

The FiM Proxy provides the ability implement so called diagnostic function inhibition proxies, enabling that software in the Application Software Clusters can access the FiM in the Host Software Cluster, to get function permissions or to set the function availability.

The specified FiM Proxy has the following underlying design principle:

In the Application Software Cluster, Function Inhibitions supported by the FiM High Proxy are configured with EcucModuleConfigurationValues for a FiM EcucModuleDef, with according FiMFIDs containers. In the Application Software Cluster and in the Host Software Cluster, the configuration of SwCluCFiMProxyFID containers map the FiMFIDs to CpSoftwareClusterServiceResources, defining each the globalResourceId and the isMandatory attribute, relevant for the corresponding Resource Entry in the Binary Manifest.

The connection between the FiM High Proxy and the FiM Low Proxy utilizes the Binary Manifest. Finally, the FiM Low Proxy invokes the FiM APIs in the Host Software Cluster in the partitions the FiM module is supporting.

The configuration of the Function Inhibition with the relationships between diagnostic events and FIDs resides in the Host Software Cluster. To adjust the intended inhibition without rebuild of the Host Software Cluster, the FiM module provides post-build configurability to large extend.

7.4.3.6.1 Enable FiM Proxy Generation

[SWS_SwCluC_03214]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00204

⌈The FiM High Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationFiM is set to HIGH_PROXY.⌋

[SWS_SwCluC_03215]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00204

⌈The FiM Low Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationFiM is set to LOW_PROXY.⌋

7.4.3.6.2 FiM High Proxy

7.4.3.6.2.1 FiM High Proxy C-API

[SWS_SwCluC_03200]

Upstream requirements: SRS_SwCluC_00210

⌈The `FiM High Proxy` shall provide the header file `FiM.h`.⌋

[SWS_SwCluC_03201]

Upstream requirements: SRS_SwCluC_00210

⌈The `FiM High Proxy` shall provide the symbolic name values for `FiMFID`s, for FiM users in the `Application Software Cluster`.⌋

[SWS_SwCluC_03202]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00202

⌈The `FiM High Proxy` shall provide the functions for `Function Inhibition`s

- `FiM_GetFunctionPermission`
- `FiM_SetFunctionAvailable`

for FiM users in the `Application Software Cluster`.⌋

[SWS_SwCluC_03217]

Upstream requirements: SRS_SwCluC_00210, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If development error detection is enabled for `FiM High Proxy`, the APIs

- `FiM_GetFunctionPermission`
- `FiM_SetFunctionAvailable`

shall report the DET error `FIM_E_UNINIT`, when `FiM High Proxy` is not yet initialized.⌋

[SWS_SwCluC_03218]

Upstream requirements: SRS_SwCluC_00210, SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350

⌈If development error detection is enabled for `FiM High Proxy`, the APIs

- `FiM_GetFunctionPermission`
- `FiM_SetFunctionAvailable`

shall report the DET error `FIM_E_FID_OUT_OF_RANGE`, if the functions are called with wrong `FID` value⌋

7.4.3.6.2.2 Service Ports

[SWS_SwCluC_03204]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00201

⌈The `FiM High Proxy` shall provide the ports belonging to a specific `Function Inhibition`

- `Func_{Name}`
- `Control_{Name}`

at the partition specific `Service Software Component`, to which the using `SwComponentPrototype` is mapped.⌋

Multiple Partitions

[SWS_SwCluC_03203]

Upstream requirements: SRS_SwCluC_00210

⌈The `FiM High Proxy` shall create one partition specific `Service Software Component` per configured `EcucPartition`, with the name `FiM_<EcucPartition shortName>`.⌋

7.4.3.6.2.3 Handling of unconnected `Function Inhibition`

[SWS_SwCluC_03205]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00213

⌈In case a `Function Inhibition` configured in the `FiM High Proxy` is not connected to any `FiM Low Proxy`, the `FiM High Proxy` shall return `E_NOT_OK` and the value `FALSE` for the `Permission`, when `FiM_GetFunctionPermission` / `GetFunctionPermission` is called.⌋

[SWS_SwCluC_03206]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00213

⌈In case a `Function Inhibition` configured in the `FiM High Proxy` is not connected to any `FiM Low Proxy`, the `FiM High Proxy` shall return `E_NOT_OK`, when `FiM_SetFunctionAvailable` / `SetFunctionAvailable` is called.⌋

7.4.3.6.2.4 Binary Manifest

The implementation of the `FiM High Proxy` may require some basic infrastructure from the `FiM Low Proxy` in order to be operational. Such generic infrastructure services can be linked between `Application Software Cluster` and `Host Software Cluster` by means of a `FiM base socket`.

[SWS_SwCluC_03210]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00212

⌈The FiM High Proxy shall create for each SwCluCFiMProxyBaseSocket a required resource entry in the Binary Manifest.⌋

[SWS_SwCluC_03211]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00212

⌈The FiM High Proxy shall create for each diagnostic function inhibition proxy - configured via SwCluCFiMProxyFID container - a required resource entry in the Binary Manifest.⌋

7.4.3.6.3 FiM Low Proxy

As shown in figure 7.10, the FiM Low Proxy shall be able to invoke APIs of the BSW modules in the Host Software Cluster, without a partition change in the Application Software Clusters. This partition change is only needed, if the BSW module does not offer a satellite on this partition. The availability of the BSW modules on dedicated EcucPartitions is a property of the Host Software Cluster.

[SWS_SwCluC_03207]

Upstream requirements: SRS_SwCluC_00210

⌈The FiM Low Proxy implementation shall describe the Function Inhibitions provided to FiM High Proxys via FunctionInhibitionNeeds, derived from the CpSoftwareClusterServiceResource.resourceNeeds.⌋

[SWS_SwCluC_03208]

Upstream requirements: SRS_SwCluC_00210

⌈The FiM Low Proxy implementation shall describe the control functions for Function Inhibitions provided to FiM High Proxies via FunctionInhibitionAvailabilityNeeds, derived from the CpSoftwareClusterServiceResource.resourceNeeds.⌋

Note: [SWS_SwCluC_03205] and [SWS_SwCluC_03206] ensures a partly automated configuration of the FiM Manager.

[SWS_SwCluC_03209]

Upstream requirements: SRS_SwCluC_00210

⌈The FiM Low Proxy shall provide, for each SwCluCFiMProxyBaseSocket, a SwCluC_FiMProxy_Init API, to initialize the FiM High Proxy of an Application Software Cluster. This shall be one element of the service resource SwCluCFiMProxyBaseSocket.⌋

7.4.3.6.3.1 Binary Manifest

The implementation of the FiM Low Proxy may provide some basic infrastructure for the FiM High Proxy in order to be operational. Such generic infrastructure services can be linked between Application Software Cluster and Host Software Cluster by means of a FiM base socket.

[SWS_SwCluC_03212]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00212

⌈The FiM Low Proxy shall create for each SwCluCFiMProxyBaseSocket a provided resource entry in the Binary Manifest.⌋

[SWS_SwCluC_03213]

Upstream requirements: SRS_SwCluC_00210, SRS_SwCluC_00212

⌈The FiM Low Proxy shall create for each diagnostic function inhibition proxy - configured via SwCluCFiMProxyFID container - a provided resource entry in the Binary Manifest.⌋

7.4.4 Error Classification

7.4.4.1 Development Errors

The development error types defined for the SwCluC NvM Proxy are listed in [SWS_SwCluC_02140].

[SWS_SwCluC_02140] Development Error Types of SwCluC NvM Proxy

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Value [hex]*
API service called with wrong parameter	NVM_E_PARAM_BLOCK_ID	0x0A
API is called with wrong parameter pointer	NVM_E_PARAM_POINTER	0x0F
API is called when NvM Proxy is not initialized yet	NVM_E_UNINIT	0x14

⌋

The development error types defined for the SwCluC Com Proxy are listed in [SWS_SwCluC_02650].

[SWS_SwCluC_02650] Development Error Types of SwCluC Com Proxy

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Value [hex]*
API service called with wrong parameter	COM_E_PARAM	0x01
API service called with a NULL pointer.	COM_E_PARAM_POINTER	0x03
Error code if any API service is called before the `Com High Proxy` module was initialized by the `Com Low Proxy`	COM_E_UNINIT	0x02

⌋

The development error types defined for the SwCluC Dcm Proxy are listed in [SWS_SwCluC_02426].

[SWS_SwCluC_02426] Development Error Types of SwCluC Dcm Proxy

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Value [hex]*
Error code, if any API service is called before the `Dcm High Proxy` module was initialized by the `Dcm Low Proxy`	DCM_E_UNINIT	0x05

⌋

The development error types defined for the SwCluC Dem Proxy are listed in [SWS_SwCluC_02750].

[SWS_SwCluC_02750] Development Error Types of SwCluC Dem Proxy

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Value [hex]*
API function called with a NULL pointer.	DEM_E_PARAM_POINTER	0x11
API function called with wrong parameter value	DEM_E_PARAM_DATA	0x12

▽

△

API function called with wrong length parameter value	DEM_E_PARAM_LENGTH	0x13
Error code, if any API service is called before the Dem High Proxy module was initialized by the Dem Low Proxy	DEM_E_UNINIT	0x20

⌋

The development error types defined for the SwCluC FiM Proxy are listed in [SWS_SwCluC_03216].

[SWS_SwCluC_03216] Development Error Types of SwCluC FiM Proxy

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Value [hex]*
Error code, if any API service is called before the FiM High Proxy module was initialized by the FiM Low Proxy	FIM_E_UNINIT	0x01
FiM_GetFunctionPermission or FiM_SetFunctionAvailable called with wrong FID	FIM_E_FID_OUT_OF_RANGE	0x02

⌋

7.4.4.2 Runtime Errors

Runtime errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.4.4.3 Production Errors

Production Errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.5 Standardized Service Resources

This section list the standardized definitions of CpSoftwareClusterService-Resource with the defined categorys and the applicable resourceNeeds.

Table 7.4 defines the meaning of the tables for CpSoftwareClusterService-Resource in this section.

Column	Description	
Category	category of the CpSoftwareClusterServiceResource referencing hte related EcucContainerValues via the resourceNeeds reference.	
Context and **Container / Parameter**	path to the standardized definition	
Mult.	Describes the multiplicity, which applies for the Container, Parameter, or Reference when it is used for the description of a CpSoftwareClusterService-Resource.resourceNeeds. This multicity can deviate from the multiplicity defind for the Standardized Module Defintion	
Description / Restrictions	Describes the multiplicity, which applies for the Container, Parameter, or Reference when it is used for the description of a CpSoftwareClusterService-Resource.resourceNeeds	
Guard Value relevant	Defines whether and how the Container, Parameter, or Reference is treated for the calculation of the Guard Value of a Resource Entry. Following list of standard definitions apply:	
	No	The Container, Parameter, or Reference is only used for information or to express a configuration request which is optionally to be fulfilled and does not impact the calculated Guard Value
	Value	The value of the Parameter is considered for the Guard Value calculation
	Choice	Applicable in case the mentioned Container is child of a choice container. The selected kind chosen Container is part of the Guard Value calculation
	from ref	Take the attributes from the referenced Container. The related Container, Parameter, or Reference will be given additionally in this table

Table 7.4: How to read Standardized Service Resource tables

7.5.1 Software Cluster Base Configuration Check

For a CpSoftwareClusterServiceResource, defined for a Software Cluster Base Configuration Check, [SWS_SwCluC_90000] applies.

[SWS_SwCluC_90000] SWCLUSTER_RES_BASE_CNF ⌈

Category	SWCLUSTER_RES_BASE_CNF			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.5: SWCLUSTER_RES_BASE_CNF

⌋

7.5.2 Cross Cluster Communication

For a `CpSoftwareClusterServiceResource`, defined for a `Cross Cluster Communication Base Socket` (see section 7.3.2.3), [SWS_SwCluC_90008] applies.

[SWS_SwCluC_90008] SWCLUSTER_RES_XCC_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_XCC_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.6: SWCLUSTER_RES_XCC_BASE_SOCKET

⌋

7.5.3 OS Proxy

For a `CpSoftwareClusterServiceResource`, defined for a `Os Base Socket` (see section 7.4.3.1.2), [SWS_SwCluC_90002] applies.

[SWS_SwCluC_90002] SWCLUSTER_RES_OS_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_OS_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.7: SWCLUSTER_RES_OS_BASE_SOCKET

⌋

For a CpSoftwareClusterServiceResource, defined for a OsApplication (see section 7.4.3.1.4), [SWS_SwCluC_90003] applies.

[SWS_SwCluC_90003] SWCLUSTER_RES_OS_APPLICATION ⌈

Category	SWCLUSTER_RES_OS_APPLICATION			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.8: SWCLUSTER_RES_OS_APPLICATION

⌋

For a CpSoftwareClusterServiceResource, defined for a OsTask (see section 7.4.3.1.4), [SWS_SwCluC_90004] applies.

[SWS_SwCluC_90004] SWCLUSTER_RES_OS_TASK ⌈

Category	SWCLUSTER_RES_OS_TASK			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
Os/OsTask	OsTaskActivation	1		Value
Os/OsTask	OsTaskPriority	1		Value
Os/OsTask	OsTaskSchedule	1		Value

Table 7.9: SWCLUSTER_RES_OS_TASK

⌋

For a CpSoftwareClusterServiceResource, defined for a OsResource (see section 7.4.3.1.4), [SWS_SwCluC_90005] applies.

[SWS_SwCluC_90005] SWCLUSTER_RES_OS_RESOURCE ⌈

Category	SWCLUSTER_RES_OS_RESOURCE			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.10: SWCLUSTER_RES_OS_RESOURCE

⌋

For a CpSoftwareClusterServiceResource, defined for a OsSpinlock (see section 7.4.3.1.4), [SWS_SwCluC_90006] applies.

[SWS_SwCluC_90006] SWCLUSTER_RES_OS_SPINLOCK ⌈

Category	SWCLUSTER_RES_OS_SPINLOCK			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.11: SWCLUSTER_RES_OS_SPINLOCK

⌋

For a `CpSoftwareClusterServiceResource`, defined for a `function dispatcher` (see section 7.4.3.1.4), [SWS_SwCluC_90008] applies.

[SWS_SwCluC_90009] SWCLUSTER_RES_OS_FNC_DISPATCHER ⌈

Category	SWCLUSTER_RES_OS_FNC_DISPATCHER			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.12: SWCLUSTER_RES_OS_FNC_DISPATCHER

⌋

7.5.4 NvM Proxy

For a `CpSoftwareClusterServiceResource`, defined for a `NvM Base Socket` (see section 7.4.3.2.5), [SWS_SwCluC_90010] applies.

[SWS_SwCluC_90010] SWCLUSTER_RES_NVM_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_NVM_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.13: SWCLUSTER_RES_NVM_BASE_SOCKET

⌋

For a `CpSoftwareClusterServiceResource`, defined for a NV Block (see section 7.4.3.2), [SWS_SwCluC_90001] applies.

[SWS_SwCluC_90001] SWCLUSTER_RES_NV_BLOCK ⌈

Category	SWCLUSTER_RES_NV_BLOCK			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
NvM/NvMBlockDescriptor	NvMBlockManagementType	1		Value
NvM/NvMBlockDescriptor	NvMBlockUseAutoValidation	1		Value
NvM/NvMBlockDescriptor	NvMBlockUseCrc	1		Value
NvM/NvMBlockDescriptor	NvMBlockUseSetRamBlock Status	1		Value
NvM/NvMBlockDescriptor	NvMBlockWriteProt	1		Value
NvM/NvMBlockDescriptor	NvMCalcRamBlockCrc	1		Value
NvM/NvMBlockDescriptor	NvMResistantToChangedSw	1		Value
NvM/NvMBlockDescriptor	NvMRomBlockNum	1		Value
NvM/NvMBlockDescriptor	NvMSelectBlockForReadAll	1		Value
NvM/NvMBlockDescriptor	NvMSelectBlockForWriteAll	1		Value
NvM/NvMBlockDescriptor	NvMWriteBlockOnce	1		Value
NvM/NvMCommon	NvMSetRamBlockStatusApi	1		Value
SwCluC/SwCluCProxies/SwCluCNvMProxy/SwCluCNvMProxyNvBlock	SwCluCNvMProxyNvBlockMaxLength	1		Value

Table 7.14: SWCLUSTER_RES_NV_BLOCK

⌋

7.5.5 Com Proxy

For a `CpSoftwareClusterServiceResource`, defined for a `Com base socket`, [SWS_SwCluC_90011] applies.

[SWS_SwCluC_90011] SWCLUSTER_RES_COM_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_COM_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.15: SWCLUSTER_RES_COM_BASE_SOCKET

⌋

7.5.6 Dcm Proxy

For a `CpSoftwareClusterServiceResource`, defined for a `Dcm base socket`, [SWS_SwCluC_90013] applies.

[SWS_SwCluC_90013] SWCLUSTER_RES_DCM_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_DCM_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.16: SWCLUSTER_RES_DCM_BASE_SOCKET

⌋

For a CpSoftwareClusterServiceResource, defined for a diagnostic data proxy, [SWS_SwCluC_90014] applies.

[SWS_SwCluC_90014] SWCLUSTER_RES_DCM_DIAGNOSTIC_DATA ⌈

Category	SWCLUSTER_RES_DCM_DIAGNOSTIC_DATA			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.17: SWCLUSTER_RES_DCM_DIAGNOSTIC_DATA

⌋

For a CpSoftwareClusterServiceResource, defined for a routine control proxy, [SWS_SwCluC_90015] applies.

[SWS_SwCluC_90015] SWCLUSTER_RES_DCM_ROUTINE_CONTROL ⌈

Category	SWCLUSTER_RES_DCM_ROUTINE_CONTROL			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.18: SWCLUSTER_RES_DCM_ROUTINE_CONTROL

⌋

For a CpSoftwareClusterServiceResource, defined for a service request notification proxy, [SWS_SwCluC_90016] applies.

[SWS_SwCluC_90016] SWCLUSTER_RES_DCM_SERVICE_REQUEST_NOTIFICATION ⌈

Category	SWCLUSTER_RES_DCM_SERVICE_REQUEST_NOTIFICATION			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.19: SWCLUSTER_RES_DCM_SERVICE_REQUEST_NOTIFICATION

⌋

7.5.7 Dem Proxy

For a `CpSoftwareClusterServiceResource`, defined for a `Dem base socket`, [SWS_SwCluC_90017] applies.

[SWS_SwCluC_90017] SWCLUSTER_RES_DEM_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_DEM_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.20: SWCLUSTER_RES_DEM_BASE_SOCKET

⌋

For a `CpSoftwareClusterServiceResource`, defined for a `diagnostic event proxy`, [SWS_SwCluC_90018] applies.

[SWS_SwCluC_90018] SWCLUSTER_RES_DEM_DIAGNOSTIC_EVENT ⌈

Category	SWCLUSTER_RES_DEM_DIAGNOSTIC_EVENT			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
Dem/DemConfigSet/DemEventParameter	DemDebounceAlgorithmClass	1	choice of debouncing without contained parameters	Choice
Dem/DemConfigSet/DemEventParameter	DemEventReportingType	1		Value

Table 7.21: SWCLUSTER_RES_DEM_DIAGNOSTIC_EVENT

⌋

7.5.8 FiM Proxy

For a CpSoftwareClusterServiceResource, defined for a FiM base socket [SWS_SwCluC_90019], applies.

[SWS_SwCluC_90019] SWCLUSTER_RES_FIM_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_FIM_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.22: SWCLUSTER_RES_FIM_BASE_SOCKET

⌋

For a CpSoftwareClusterServiceResource, defined for a diagnostic function inhibition proxy, [SWS_SwCluC_90020] applies.

[SWS_SwCluC_90020] SWCLUSTER_RES_FIM_FID ⌈

Category	SWCLUSTER_RES_FIM_FID			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.23: SWCLUSTER_RES_FIM_FID

⌋

7.5.9 WdgM

For a CpSoftwareClusterServiceResource defined for a WdgM base socket [SWS_SwCluC_90021] applies.

[SWS_SwCluC_90021] SWCLUSTER_RES_WDGM_BASE_SOCKET ⌈

Category	SWCLUSTER_RES_WDGM_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.24: SWCLUSTER_RES_WDGM_BASE_SOCKET

⌋

Please note the definition of WdgMBaseSocket in document [19]

For a CpSoftwareClusterServiceResource defined for a cross cluster transition proxy [SWS_SwCluC_90022] applies.

[SWS_SwCluC_90022] SWCLUSTER_RES_WDGM_TRANSITION ⌈

Category	SWCLUSTER_RES_WDGM_TRANSITION			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.25: SWCLUSTER_RES_WDGM_TRANSITION

⌋

Please note the definition of WdgMTransitionProxy in document [19]

7.6 Common Functions

This section describes common functionality of the SwCluC module.

7.6.1 Error Classification

7.6.1.1 Development Errors

The general development error types defined for the SwCluC module are listed in [SWS_SwCluC_09001].

[SWS_SwCluC_09001] Definition of development errors in module SwCluC

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Error value*
Error code, if API is called with wrong parameter pointer	SWCLUC_E_PARAM_POINTER	0x01

⌋

8 API specification

8.1 Imported types

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

[SWS_SwCluC_10012] Definition of imported datatypes of module SwCluC

Upstream requirements: SRS_SwCluC_00202, SRS_SwCluC_00211

⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	CbkHandleIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
Dem	Rte_Dem_Type.h	Dem_DTCFormatType
	Rte_Dem_Type.h	Dem_DebounceResetStatusType
	Rte_Dem_Type.h	Dem_DebouncingStateType
	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
	Rte_Dem_Type.h	Dem_MonitorDataType
	Rte_Dem_Type.h	Dem_MonitorStatusType
	Rte_Dem_Type.h	Dem_UdsStatusByteType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
	Rte_NvM_Type.h	NvM_RequestResultType
Std	Std_Types.h	Std_ReturnType

⌋

Please note that the `Software Cluster Connection` additionally utilizes the Platform Types (as defined in document [20]) and Standard Types (as defined in document [21]).

8.2 Type definitions

8.2.1 Binary Manifest

8.2.1.1 SwCluC_BManif_SwClusterIdType

[SWS_SwCluC_10000] Definition of datatype SwCluC_BManif_SwClusterIdType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_SwClusterIdType
Kind	Type
Derived from	uint8
Description	Type of the Software Cluster's ID value
Available via	SwCluC_BManif.h

⌋

8.2.1.2 SwCluC_BManif_MachineIdType

[SWS_SwCluC_10001] Definition of datatype SwCluC_BManif_MachineIdType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_MachineIdType
Kind	Type
Derived from	uint8
Description	Type for the global ID value of a resource
Available via	SwCluC_BManif.h

⌋

8.2.1.3 SwCluC_BManif_ConCtrlType

[SWS_SwCluC_10002] Definition of datatype SwCluC_BManif_ConCtrlType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_ConCtrlType		
Kind	Type		
Derived from	uint16		
Range	SWCLUC_BMANIF_DISABLE_ON_ECU_CONNECTION	0x8000	The connection of Software Clusters on ECU is disabled for this machine.

▽

△

Description	Type to code control flags for the connection process. Bit 0..14 are reserved.
Available via	SwCluC_BManif.h

⌋

8.2.1.4 SwCluC_BManif_ResourcePropertiesType

[SWS_SwCluC_10003] Definition of datatype SwCluC_BManif_ResourcePropertiesType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_ResourcePropertiesType			
Kind	Bitfield			
Derived from	uint8			
Elements	***Kind***	***Name***	***Mask***	***Description***
	bit	SWCLUC_BMANIF_PROVIDED_RESOURCE	0x80	the resource is provided by the Software Cluster
	bit	SWCLUC_BMANIF_MANDATORY_RESOURCE	0x40	the required resource is mandatory
Description	Type to code the properties of a resource. Bit 0..5 are reserved. In case none of the bits are set the value 0 is valid.			
Available via	SwCluC_BManif.h			

⌋

8.2.1.5 SwCluC_BManif_ResourceTypeIdType

[SWS_SwCluC_10004] Definition of datatype SwCluC_BManif_ResourceTypeIdType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_ResourceTypeIdType
Kind	Type
Derived from	uint8
Description	Type to code the type of a resource
Available via	SwCluC_BManif.h

⌋

8.2.1.6 SwCluC_BManif_GlobalResourceIdType

[SWS_SwCluC_10005] Definition of datatype SwCluC_BManif_GlobalResourceIdType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_GlobalResourceIdType
Kind	Type
Derived from	uint32
Description	Type for the global ID value of a resource
Available via	SwCluC_BManif.h

⌋

8.2.1.7 SwCluC_BManif_ResourceGuardValueType

[SWS_SwCluC_10006] Definition of datatype SwCluC_BManif_ResourceGuardValueType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_ResourceGuardValueType
Kind	Type
Derived from	uint32
Description	Type for the guard value of a resource
Available via	SwCluC_BManif.h

⌋

8.2.1.8 SwCluC_BManif_TableIndexType

[SWS_SwCluC_10007] Definition of datatype SwCluC_BManif_TableIndexType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_TableIndexType
Kind	Type
Derived from	uint16
Description	Index type to address the immutable and modifiable interface table
Available via	SwCluC_BManif.h

⌋

8.2.1.9 SwCluC_BManif_HandleIndexType

[SWS_SwCluC_10008] Definition of datatype SwCluC_BManif_HandleIndexType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_HandleIndexType
Kind	Type
Derived from	uint8
Description	Index type to address a set of handles in the immutable and modifiable interface table
Available via	SwCluC_BManif.h

⌋

8.2.1.10 SwCluC_BManif_VoidFncPtrType

[SWS_SwCluC_10009] Definition of datatype SwCluC_BManif_VoidFncPtrType

Upstream requirements: SRS_SwCluC_00006

⌈

Name	SwCluC_BManif_VoidFncPtrType	
Kind	Function Pointer	
Syntax	`int (*SwCluC_BManif_VoidFncPtrType) ( void )`	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	int	–
Description	Generic function pointer type according C default type for functions	
Available via	SwCluC_BManif.h	

⌋

8.2.1.11 SwCluC_BManif_HandleType

[SWS_SwCluC_10010] Definition of datatype SwCluC_BManif_HandleType

Upstream requirements: SRS_SwCluC_00006

⌈

Name	SwCluC_BManif_HandleType	
Kind	Union	
Elements	dptr	
	Type	void*

▽

△

	Comment	Handle as pointer to variable
	val	
	Type	uint32
	Comment	Handle as value
	fptr	
	Type	SwCluC_BManif_VoidFncPtrType
	Comment	Handle as pointer to function
Description	Type of a handle in the interface table.	
Available via	SwCluC_BManif.h	

⌋

8.2.1.12 SwCluC_BManif_HeaderType

[SWS_SwCluC_10011] Definition of datatype SwCluC_BManif_HeaderType

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Name	SwCluC_BManif_HeaderType	
Kind	Structure	
Elements	Preamble	
	Type	uint64
	Comment	magic pattern of Manifest Header begin
	ManifestMajorVersion	
	Type	uint8
	Comment	version of the Binary Manifest Layout
	ManifestMinorVersion	
	Type	uint8
	Comment	version of the Binary Manifest Layout
	SwClusterId	
	Type	SwCluC_BManif_SwClusterIdType
	Comment	Software Cluster Id
	MachineId	
	Type	SwCluC_BManif_MachineIdType
	Comment	Machine Id
	SwClusterType	
	Type	uint8
	Comment	kind of Software Cluster
	Reserved1	
	Type	uint8
	Comment	reserved
	Reserved2	
	Type	uint8

▽

△

	Comment	reserved
	Reserved3	
	Type	uint8
	Comment	reserved
	ConnectorControlFlags	
	Type	SwCluC_BManif_ConCtrlType
	Comment	Control flags for Software Cluster connector
	NoOfInterfaceDescriptors	
	Type	uint16
	Comment	Number of Interface Descriptors
	NoOfOfferedInterfaceHandles	
	Type	uint16
	Comment	Number of Handles in the Offered Interface
	NoOfSubscribedInterfaceHandles	
	Type	uint16
	Comment	Number of Handles in the Subscribed Interface
	ImmutableTablesChecksumPtr	
	Type	const uint32*
	Comment	Pointer to checksum about immutable descriptor and interface tables
	SubscribedInterfaceValidityMarkerPtr	
	Type	const uint32*
	Comment	Pointer to validity marker indicating whether all subscribed tables are written
	ResourcePropertiesDescriptorColumnPtr	
	Type	const SwCluC_BManif_ResourcePropertiesType*
	Comment	Pointer to descriptor column for resource properties
	ResourceTypeDescriptorColumnPtr	
	Type	const SwCluC_BManif_ResourceTypeIdType*
	Comment	Pointer to descriptor column for resource type Ids
	GlobalResourceIdDescriptorColumnPtr	
	Type	const SwCluC_BManif_GlobalResourceIdType*
	Comment	Pointer to descriptor column for global resource Ids
	ResourceGuardValueDescriptorColumnPtr	
	Type	const SwCluC_BManif_ResourceGuardValueType*
	Comment	Pointer to descriptor column for guard values
	OfferedInterfaceIndexDescriptorColumnPtr	
	Type	const SwCluC_BManif_TableIndexType*
	Comment	Pointer to descriptor column for offered interface column index
	OfferedInterfaceNoOfHandlesDescriptorColumnPtr	
	Type	const SwCluC_BManif_HandleIndexType*
	Comment	Pointer to descriptor column for number of handles in offered interface column
	SubscribedInterfaceIndexDescriptorColumnPtr	
	Type	const SwCluC_BManif_TableIndexType*

▽

△

	Comment	Pointer to descriptor column for subscribed interface column index
	SubscribedInterfaceNoOfHandlesDescriptorColumnPtr	
	Type	const SwCluC_BManif_HandleIndexType*
	Comment	Pointer to descriptor column for number of handles in subscribed interface column
	SubscribedInterfaceNoOfHandleSetsDescriptorColumnPtr	
	Type	const SwCluC_BManif_HandleIndexType*
	Comment	Pointer to descriptor column for number of handle sets in subscribed interface column
	OfferedInterfaceHandleColumnPtr	
	Type	const SwCluC_BManif_HandleType*
	Comment	Pointer to interface column for offered handles
	SubscribedInterfaceDefaultHandleColumnPtr	
	Type	const SwCluC_BManif_HandleType*
	Comment	Pointer to default interface column for subscribed handles
	SubscribedInterfaceHandleColumnPtr	
	Type	const SwCluC_BManif_HandleType*
	Comment	Pointer to interface column for subscribed handles
	SubscribedInterfaceConnectedSwClusterIdColumnPtr	
	Type	const SwCluC_BManif_SwClusterIdType*
	Comment	Pointer to interface column for connected cluster Id values
Description	–	
Available via	SwCluC_BManif.h	

⌋

8.2.2 Cross Cluster Communication

The `Cross Cluster Communication` implements the types of a `Cross Software Cluster Communication Plug-In` as specified in document [4].

8.2.3 ProxyModules

The `Proxy Modules` partly implement the types of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 BinaryManifest

8.3.1.1 API Principles

None of the functions defined for the Binary Manifest access are utilizing the Std_ReturnType to provide runtime errors. Since the Binary Manifest itself is implemented by a set of const variables, which need to be located to Flash ROM, all the failure situations caused by a temporary unavailability of the Binary Manifest can be excluded. Instead, it needs to be considered that the Binary Manifest is involved very frequently, since it is needed to access any resource from other Software Clusters, including the Host Software Cluster.

8.3.1.1.1 Named versus indexed handle access

The APIs for the Binary Manifest offer basically two different kinds of access.

The functions [SWS_SwCluC_10020], [SWS_SwCluC_10021], and [SWS_SwCluC_10032] provide an entire named access to the handle of are Resource Entry in a Resource Entry Group. This kind of access is useful, if the using implementation requires a named access on a specific channel. For instance the callbacks of NvM are implemented NvBlock specific. In such a NvBlock specific callback the related handles to call the High Proxy could be easily accessed via name.

In opposite the functions [SWS_SwCluC_10022], [SWS_SwCluC_10023], and [SWS_SwCluC_10033] provide an access to a handle of a Resource Entry in a Resource Entry Group, where the Resource Entry is addressed by index. This kind of access is useful, if the using implementation requires indexed based access on channels. Usually service APIs of the High Proxies select the channel by an ID value. E.g. the NvBlock gets selected by a BlockId parameter. Since the order of Resource Entries in a Resource Entry Group is configurable, the proxy implementation can ensure that the 'channel id' matches to the resIndex values. Hence, the passed 'channel id' values can be used without further conversion to get the relevant handles in the Binary Manifest.

8.3.1.1.2 Implementation aspects

When an implementation of the Software Cluster Connection or a CDD utilizes the Binary Manifest, it is not expected that the usage of the SwCluC_BManif_GetHandle, SwCluC_BManif_GetConSwClusterId, and SwCluC_BManif_GetNoOfHandleSets APIs result in real function calls. From the implementation perspective of the Binary Manifest, this should never be required, since accesses to handles or Software Cluster Ids are accesses to primitive constant variables.

[SWS_SwCluC_00057]

Upstream requirements: SRS_SwCluC_00012

⌈The functions SwCluC_BManif_GetHandle, SwCluC_BManif_GetConSwClusterId, and SwCluC_BManif_GetNoOfHandleSets, according to [SWS_SwCluC_10020], [SWS_SwCluC_10021], [SWS_SwCluC_10022], [SWS_SwCluC_10023], [SWS_SwCluC_10032], and [SWS_SwCluC_10033], shall each be implemented as a C-function-like macro.⌋

[SWS_SwCluC_00058]

Upstream requirements: SRS_SwCluC_00012

⌈<Resource Entry Group> is the shortName of either the SwCluCBManifProvideResourceEntryGroup or SwCluCBManifRequireResourceEntryGroup.⌋

[SWS_SwCluC_00059]

Upstream requirements: SRS_SwCluC_00012

⌈<Resource Entry> is the shortName of either the SwCluCBManifProvideResourceEntry or SwCluCBManifRequireResourceEntry.⌋

[SWS_SwCluC_00060]

Upstream requirements: SRS_SwCluC_00012

⌈<Handle> is the shortName of either the SwCluCBManifProvideHandle or SwCluCBManifNotifierHandle of the SwCluCBManifResourceType, referenced by SwCluCBManifProvideResourceEntryGroup or SwCluCBManifRequireResourceEntryGroup.⌋

[SWS_SwCluC_00061]

Upstream requirements: SRS_SwCluC_00012

⌈<handleType> is either

- the referenced ImplementationDataType, if the SwCluCBManifHandleImpleTypeRef is defined for a Provide Handle of Notifier Handle. OR
- uint32, if the SwCluCBManifHandleImpleTypeRef is not defined, and SwCluCBManifNativeHandleType is set to VALUE OR
- void *, if the SwCluCBManifHandleImpleTypeRef is not defined, and SwCluCBManifNativeHandleType is set to DATA_REFERENCE OR
- SwCluC_BManif_VoidFncPtrType, if the SwCluCBManifHandleImpleTypeRef is not defined, and SwCluCBManifNativeHandleType is set to FUNCTION_REFERENCE

⌋

[SWS_SwCluC_CONSTR_00091]

Upstream requirements: SRS_SwCluC_00014

⌈In a set of SwCluCBManifProvideHandle in a SwCluCBManifResourceType it is not permitted to set SwCluCBManifIsOptional to false, if any preceding SwCluCBManifProvideHandle has already set SwCluCBManifIsOptional to true.⌋

[SWS_SwCluC_CONSTR_00092]

Upstream requirements: SRS_SwCluC_00014

⌈In a set of SwCluCBManifNotifierHandle in a SwCluCBManifResourceType it is not permitted to set SwCluCBManifIsOptional to false, if any preceding SwCluCBManifNotifierHandle has already set SwCluCBManifIsOptional to true.⌋

Rationale: A Resource Type with optional handles has to define the optional ones after the non-optional ones.

[SWS_SwCluC_CONSTR_00093]

Upstream requirements: SRS_SwCluC_00014

⌈In a set of SwCluCBManifProvideHandle in a SwCluCBManifResourceType at most one optional handle shall exist.⌋

[SWS_SwCluC_CONSTR_00094]

Upstream requirements: SRS_SwCluC_00014

⌈In a set of SwCluCBManifNotifierHandle in a SwCluCBManifResourceType at most one optional handle shall exist.⌋

Rationale: If more than one optional handle exist, the mapping between the filled handles and the declared handles in the resource type becomes ambigous. If an implementation requires to vary the number of exchanged handles on a more fine grained level, those can be put into implementation specifc structure behind the optional handle. But the minor configurtion with the typical highest optimization need can still work without double indirection.

8.3.1.2 SwCluC_BManif_GetHandle

[SWS_SwCluC_10020] Definition of API function SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<ResourceEntry>_<Handle>

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Service Name	SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<ResourceEntry>_<Handle>	
Syntax	`<handleType> SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<ResourceEntry>_<Handle> ( SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_ SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported
Parameters (inout)	None	
Parameters (out)	None	
Return value	<handleType>	Pointer or value stored in Binary Manifest for this handle
Description	Returns a handle of a Resource Entry in a Resource Entry Group	
Available via	SwCluC_BManif.h	

⌋

[SWS_SwCluC_00062]

Upstream requirements: SRS_SwCluC_00006

⌈The function `SwCluC_BManif_GetHandle` according to [SWS_SwCluC_10020] shall exist for each `SwCluCBManifProvideHandle` and `SwCluCBManifNotifierHandle` defined for each `SwCluCBManifProvideResourceEntry` and `SwCluCBManifRequireResourceEntry` in a `SwCluCBManifProvideResourceEntryGroup` and `SwCluCBManifRequireResourceEntryGroup`.⌋

8.3.1.3 SwCluC_BManif_GetHandle

[SWS_SwCluC_10022] Definition of API function SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<Handle>

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Service Name	SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<Handle>	
Syntax	`<handleType> SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<Handle> ( SwCluC_BManif_TableIndexType resIndex, SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	resIndex	Index of the resource entry in a Resource Entry Group in the range 0.. SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group> -1.
	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_ SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported
Parameters (inout)	None	
Parameters (out)	None	
Return value	<handleType>	Id of the connected Software Cluster
Description	Returns a handle of a Resource Entry in a Resource Entry Group	
Available via	SwCluC_BManif.h	

⌋

[SWS_SwCluC_00064]

Upstream requirements: SRS_SwCluC_00006

⌈The function `SwCluC_BManif_GetHandle` according to [SWS_SwCluC_10022] shall exist for each `SwCluCBManifProvideHandle` and `SwCluCBManifNotifierHandle` defined for each `SwCluCBManifProvideResourceEntryGroup` and `SwCluCBManifRequireResourceEntryGroup`.⌋

8.3.1.4 SwCluC_BManif_GetConSwClusterId

[SWS_SwCluC_10021] Definition of API function SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<ResourceEntry>_<Handle>

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310, SRS_SwCluC_00008

⌈

Service Name	SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<ResourceEntry>_<Handle>	
Syntax	`SwCluC_BManif_SwClusterIdType SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<ResourceEntry>_<Handle> ( SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_SwClusterIdType	Id of the connected Software Cluster
Description	Returns the Id of the connected Software Cluster for a Notifier Handle of a Provide Resource Entry or a Provide Handle of a Require Resource Entry	
Available via	SwCluC_BManif.h	

⌋

[SWS_SwCluC_00063]

Upstream requirements: SRS_SwCluC_00006

⌈The function `SwCluC_BManif_GetConSwClusterId` according to [SWS_SwCluC_10021] shall exist

- for each `SwCluCBManifNotifierHandle` defined for each `SwCluCBManifProvideResourceEntry` in a `SwCluCBManifProvideResourceEntryGroup`

  AND

- for each `SwCluCBManifProvideHandle` defined for each `SwCluCBManifRequireResourceEntry` in a `SwCluCBManifRequireResourceEntryGroup`.

⌋

8.3.1.5 SwCluC_BManif_GetConSwClusterId

[SWS_SwCluC_10023] Definition of API function SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<Handle>

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310, SRS_SwCluC_00008

⌈

Service Name	SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<Handle>	
Syntax	`SwCluC_BManif_SwClusterIdType SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<Handle> ( SwCluC_BManif_TableIndexType resIndex, SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	resIndex	Index of the resource entry in a Resource Entry Group in the range 0.. SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group> -1.
	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported.
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_SwClusterIdType	Id of the connected Software Cluster
Description	Returns the Id of the connected Software Cluster for a Notifier Handle of a Provide Resource Entry or a Provide Handle of a Require Resource Entry	
Available via	SwCluC_BManif.h	

⌋

[SWS_SwCluC_00065]

Upstream requirements: SRS_SwCluC_00006

⌈The function `SwCluC_BManif_GetConSwClusterId` according to [SWS_SwCluC_10023] shall exist

- for each `SwCluCBManifNotifierHandle` defined for a `SwCluCBManifProvideResourceEntryGroup`

  AND

- for each `SwCluCBManifProvideHandle` defined for a `SwCluCBManifRequireResourceEntryGroup`.

⌋

8.3.1.6 SwCluC_BManif_GetNoOfHandleSets

[SWS_SwCluC_10032] Definition of API function SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>_<Resource Entry>

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Service Name	SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>_<Resource Entry>	
Syntax	`SwCluC_BManif_HandleIndexType SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>_<Resource Entry> ( void )`	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_HandleIndexType	Number of actually used - and thereby connected - handle sets
Description	Returns the number of actually used - and thereby connected - handle sets	
Available via	SwCluC_BManif.h	

⌋

[SWS_SwCluC_00066]

Upstream requirements: SRS_SwCluC_00006

⌈The function `SwCluC_BManif_GetNoOfHandleSets` according to [SWS_SwCluC_10032] shall exist for each `SwCluCBManifProvideResourceEntry` in a `SwCluCBManifProvideResourceEntryGroup` where the parameter `SwCluCBManifMultipleNotifierSupport` is set to `MULTIPLE_NOTIFIER_SETS`.⌋

8.3.1.7 SwCluC_BManif_GetNoOfHandleSets

[SWS_SwCluC_10033] Definition of API function SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Service Name	SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>
Syntax	`SwCluC_BManif_HandleIndexType SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group> ( SwCluC_BManif_TableIndexType resIndex )`
Service ID [hex]	0x15
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant	
Parameters (in)	resIndex	Index of the resource entry in a Resource Entry Group in the range 0.. SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group> -1.
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_HandleIndexType	Number of actually used - and thereby connected - handle sets
Description	Returns the number of actually used - and thereby connected - handle sets.	
Available via	SwCluC_BManif.h	

⌋

[SWS_SwCluC_00067]

Upstream requirements: SRS_SwCluC_00006

⌈The function `SwCluC_BManif_GetNoOfHandleSets` according to [SWS_SwCluC_10033] shall exist for each `SwCluCBManifProvideResourceEntryGroup` where the parameter `SwCluCBManifMultipleNotifierSupport` is set to `MULTIPLE_NOTIFIER_SETS`.⌋

8.3.1.8 SwCluC_BManif_GetValidityMarker

[SWS_SwCluC_10034] Definition of API function SwCluC_BManif_GetValidityMarker

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00310

⌈

Service Name	SwCluC_BManif_GetValidityMarker	
Syntax	`uint32 SwCluC_BManif_GetValidityMarker ( void )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint32	Value of the Subscribed Interface Validity Marker
Description	Returns the value of the Subscribed Interface Validity Marker.	
Available via	SwCluC_BManif.h	

⌋

8.3.1.9 SWCLUC_BMANIF_NO_OF_ENTRIES

[SWS_SwCluC_01000]

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00441

⌈
Symbolic name: SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group>
Value: Number of Resource Entrys in the according Resource Entry Group
Comments: n.a.⌋

[SWS_SwCluC_00068]

Upstream requirements: SRS_SwCluC_00006

⌈The definition SWCLUC_BMANIF_NO_OF_ENTRIES according to [SWS_SwCluC_01000] shall exist for each SwCluCBManifProvideResourceEntryGroup.⌋

8.3.1.10 SWCLUC_BMANIF_NO_OF_NOTIFIER_SETS

[SWS_SwCluC_01002]

Upstream requirements: SRS_SwCluC_00006, SRS_BSW_00441

⌈
Symbolic name: SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_SETS_
<Resource Entry Group>_<Resource Entry>
Value: Maximum number of notifier sets for the according Provide Resource Entry
Comments: n.a.⌋

[SWS_SwCluC_00069]

Upstream requirements: SRS_SwCluC_00006

⌈The definition SWCLUC_BMANIF_NO_OF_NOTIFIER_SETS according to [SWS_SwCluC_01002] shall exist for each SwCluCBManifProvideResourceEntry in a SwCluCBManifProvideResourceEntryGroup where the parameter SwCluCBManifMultipleNotifierSupport is set to MULTIPLE_NOTIFIER_SETS.⌋

8.3.2 Cross Cluster Communication

In addition to the interfaces listed in this section, the Cross Cluster Communication implements the interfaces of a Cross Software Cluster Communication Plug-In as specified in document [4].

8.3.2.1 SwCluC_Xcc_Init1

[SWS_SwCluC_11000] Definition of API function SwCluC_Xcc_Init1

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00108, SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_Xcc_Init1
Syntax	`void SwCluC_Xcc_Init1 ( void )`
Service ID [hex]	0x20
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_Xcc_Init is intended to allocate and initialize system resources used by the Software Cluster Connection. It initializes interface memory which might be needed to be read by other Software Clusters but it is not allowed to do any read access to memory of any another Software Cluster.
Available via	SwCluC.h

⌋

8.3.2.2 SwCluC_Xcc_Init2

[SWS_SwCluC_11001] Definition of API function SwCluC_Xcc_Init2

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00108, SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_Xcc_Init2
Syntax	`void SwCluC_Xcc_Init2 ( void )`
Service ID [hex]	0x21
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_Xcc_Init2 is intended to allocate and initialize system resources used by the Software Cluster Connection. It can execute initializations which require a read access to interface memory of another Software Clusters.
Available via	SwCluC.h

⌋

8.3.3 Proxy Modules

The Proxy Modules partly implement the interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

Following functions are provided in addition:

8.3.3.1 Os Proxy

8.3.3.1.1 SwCluC_OsProxy_Init

[SWS_SwCluC_12000] Definition of API function SwCluC_OsProxy_Init_<SwCluCOsProxyOsBaseSocket>

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_OsProxy_Init_<SwCluCOsProxyOsBaseSocket>
Syntax	`void SwCluC_OsProxy_Init_<SwCluCOsProxyOsBaseSocket> ( void )`
Service ID [hex]	0x30
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_OsProxy_Init is intended to allocate and initialize system resources used by the Os Proxy linked to this specific socked
Available via	SwCluC.h

⌋

The name part <SwCluCOsProxyOsBaseSocket> is the shortName of the SwCluCOsProxyOsBaseSocket.

8.3.3.2 NvM Proxy

8.3.3.2.1 SwCluC_NvMProxy_Init

[SWS_SwCluC_12100] Definition of API function SwCluC_NvMProxy_Init_<SwCluCNvMBaseSocket>

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_NvMProxy_Init_<SwCluCNvMBaseSocket>
Syntax	`void SwCluC_NvMProxy_Init_<SwCluCNvMBaseSocket> ( void )`
Service ID [hex]	0x31
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_NvMProxy_Init is intended to allocate and initialize system resources used by the NvMProxy linked to this specific socked.
Available via	SwCluC.h

⌋

The name part `<SwCluCNvMBaseSocket>` is the `shortName` of the `SwCluCNvMBaseSocket`.

8.3.3.2.2 SwCluC_NvMProxy_MultiBlockReadAllJobFinished

[SWS_SwCluC_91003] Definition of API function SwCluC_NvMProxy_MultiBlockReadAllJobFinished ⌈

Service Name	SwCluC_NvMProxy_MultiBlockReadAllJobFinished
Syntax	`void SwCluC_NvMProxy_MultiBlockReadAllJobFinished ( void )`
Service ID [hex]	0x47
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This function informs the NvM proxy of the SwCluC about the end of the NvM_ReadAll job.
Available via	SwCluC.h

⌋

8.3.3.3 Com Proxy

8.3.3.3.1 SwCluC_ComProxy_Init

[SWS_SwCluC_13001] Definition of API function SwCluC_ComProxy_Init_<SwCluCComBaseSocket>

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_ComProxy_Init_<SwCluCComBaseSocket>

Syntax	`void SwCluC_ComProxy_Init_<SwCluCComBaseSocket> (`
	`  void`
	`)`

Service ID [hex]	0x33
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_ComProxy_Init is intended to allocate and initialize system resources used by the ComProxy linked to this specific socked.
Available via	SwCluC.h

⌋

8.3.3.3.2 Callback APIs to handle ComSignals / ComSignalGroups

8.3.3.3.2.1 SwCluC_ComProxy_CbkRxAck

[SWS_SwCluC_13002] Definition of callback function SwCluC_ComProxy_<BS>_CbkRxAck

Upstream requirements: SRS_SwCluC_00211

⌈

Service Name	SwCluC_ComProxy_<BS>_CbkRxAck	

Syntax	`void SwCluC_ComProxy_<BS>_CbkRxAck (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Service ID [hex]	0x37	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	This callback function indicates that the signal of the primitive data item/event is ready for reception. It is an implementation of a <ComUser_CbkRxAck> callback.
Available via	SwCluC_ComProxy_Cbk.h

⌋

8.3.3.3.2.2 SwCluC_ComProxy_CbkInv

[SWS_SwCluC_13003] Definition of callback function SwCluC_ComProxy_<BS>_CbkInv

Upstream requirements: SRS_SwCluC_00211

⌈

Service Name	SwCluC_ComProxy_<BS>_CbkInv	
Syntax	`void SwCluC_ComProxy_<BS>_CbkInv ( CbkHandleIdType ComUserCbkHandleId )`	
Service ID [hex]	0x38	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback function indicates that COM has received a signal and parsed it as "invalid". It is an implementation of a <ComUser_CbkInv> callback.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.2.3 SwCluC_ComProxy_CbkRxTOut

[SWS_SwCluC_13004] Definition of callback function SwCluC_ComProxy_<BS>_CbkRxTOut

Upstream requirements: SRS_SwCluC_00211

⌈

Service Name	SwCluC_ComProxy_<BS>_CbkRxTOut
Syntax	`void SwCluC_ComProxy_<BS>_CbkRxTOut ( CbkHandleIdType ComUserCbkHandleId )`
Service ID [hex]	0x39
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	ComUserCbkHandleId	Com user callback handle Id
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback function indicates that the aliveTimeout after the last successful reception of the signal of the data item/event has expired (data element outdated). It is an implementation of a <ComUser_CbkRxTOut> callback.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.2.4 SwCluC_ComProxy_CbkTxAck

[SWS_SwCluC_13005] Definition of callback function SwCluC_ComProxy_<BS>_CbkTxAck

Upstream requirements: SRS_SwCluC_00211

⌈

Service Name	SwCluC_ComProxy_<BS>_CbkTxAck	
Syntax	`void SwCluC_ComProxy_<BS>_CbkTxAck ( CbkHandleIdType ComUserCbkHandleId )`	
Service ID [hex]	0x3a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback function indicates that the signal of the data item/event is already handed over by COM to the PDU router. It is an implementation of a <ComUser_CbkTxAck> callback.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.2.5 SwCluC_ComProxy_CbkTxErr

[SWS_SwCluC_13006] Definition of callback function SwCluC_ComProxy_<BS>_CbkTxErr

Upstream requirements: SRS_SwCluC_00211

⌈

Service Name	SwCluC_ComProxy_<BS>_CbkTxErr	

Syntax	`void SwCluC_ComProxy_<BS>_CbkTxErr (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Service ID [hex]	0x3b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback function indicates that an error occurred when the signal of the data item/event was handed over by COM to the PDU router. It is an implementation of a <ComUser_CbkTx Err> callback.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.2.6 SwCluC_ComProxy_CbkTxTOut

[SWS_SwCluC_13007] Definition of callback function SwCluC_ComProxy_<BS>_CbkTxTOut

Upstream requirements: SRS_SwCluC_00211

⌈

Service Name	SwCluC_ComProxy_<BS>_CbkTxTOut	

Syntax	`void SwCluC_ComProxy_<BS>_CbkTxTOut (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Service ID [hex]	0x3c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback function indicates that the timeout of TransmissionAcknowledgementRequest for sending the signal of the data item/event has expired. It is an implementation of a <ComUser_ CbkTxTOut> callback.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3 Callback APIs to handle ComIPdus

8.3.3.3.3.1 SwCluC_ComProxy_ComCbkCopyRxData

[SWS_SwCluC_91004] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkCopyRxData

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13008

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkCopyRxData	
Syntax	`BufReq_ReturnType SwCluC_ComProxy_<BS>_ComCbkCopyRxData ( CbkHandleIdType ComUserCbkHandleId, const PduInfoType* info, PduLengthType* bufferSizePtr )`	
Service ID [hex]	0x3d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the received I-PDU
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data copied successfully `BUFREQ_E_NOT_OK`: Data was not copied because an error occurred.
Description	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.2 SwCluC_ComProxy_ComCbkCopyTxData

[SWS_SwCluC_91005] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkCopyTxData

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13009

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkCopyTxData	
Syntax	`BufReq_ReturnType SwCluC_ComProxy_<BS>_ComCbkCopyTxData ( CbkHandleIdType ComUserCbkHandleId, const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr )`	
Service ID [hex]	0x3e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the Sdu DataPtr may be a NULL_PTR.
	retry	Will not be handled by Com and its upper layer.
Parameters (inout)	None	
Parameters (out)	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data has been copied to the transmit buffer completely as requested. `BUFREQ_E_BUSY`: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. `BUFREQ_E_NOT_OK`: Data has not been copied. Request failed.
Description	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.3 SwCluC_ComProxy_ComCbkRxIndication

[SWS_SwCluC_91006] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkRxIndication

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13010

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkRxIndication	

Syntax	`void SwCluC_ComProxy_<BS>_ComCbkRxIndication (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID [hex]	0x3f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to 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 a lower layer communication interface module.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.4 SwCluC_ComProxy_ComCbkStartOfReception

[SWS_SwCluC_91007] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkStartOfReception

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13015

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkStartOfReception	

Syntax	`BufReq_ReturnType SwCluC_ComProxy_<BS>_ComCbkStartOfReception (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  const PduInfoType* info,`
	`  PduLengthType TpSduLength,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the I-PDU.

▽

△

	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. `BUFREQ_E_NOT_OK`: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. `BUFREQ_E_OVFL`: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.5 SwCluC_ComProxy_ComCbkTpRxIndication

[SWS_SwCluC_91008] Definition of callback function SwCluC_Com Proxy_<BS>_ComCbkTpRxIndication

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13011

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkTpRxIndication	

Syntax	`void SwCluC_ComProxy_<BS>_ComCbkTpRxIndication (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  Std_ReturnType result`
	`)`	

Service ID [hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same sn, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the received I-PDU
	result	Result of the reception.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.6 SwCluC_ComProxy_ComCbkTriggerTransmit

[SWS_SwCluC_91009] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkTriggerTransmit

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13012

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkTriggerTransmit	
Syntax	`Std_ReturnType SwCluC_ComProxy_<BS>_ComCbkTriggerTransmit ( CbkHandleIdType ComUserCbkHandleId, PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x42	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the ID of the SDU that is requested to be transmitted
	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 (inout)	None	
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	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.7 SwCluC_ComProxy_ComCbkTxConfirmation

[SWS_SwCluC_91010] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkTxConfirmation

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13013

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkTxConfirmation
Syntax	`void SwCluC_ComProxy_<BS>_ComCbkTxConfirmation ( CbkHandleIdType ComUserCbkHandleId, Std_ReturnType result )`

▽

△

Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the PDU that has been transmitted
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

8.3.3.3.3.8 SwCluC_ComProxy_ComCbkTpTxConfirmation

[SWS_SwCluC_91011] Definition of callback function SwCluC_ComProxy_<BS>_ComCbkTpTxConfirmation

Upstream requirements: SRS_SwCluC_00211
Previous identifiers: SWS_SwCluC_13014

⌈

Service Name	SwCluC_ComProxy_<BS>_ComCbkTpTxConfirmation	
Syntax	`void SwCluC_ComProxy_<BS>_ComCbkTpTxConfirmation ( CbkHandleIdType ComUserCbkHandleId, Std_ReturnType result )`	
Service ID [hex]	0x44	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the transmitted I-PDU
	result	E_OK - transmission successful E_NOT_OK - transmission not successful
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called after a Signal has been transmitted via the TP-API on its network.	
Available via	SwCluC_ComProxy_Cbk.h	

⌋

The name part `<SwCluCComProxyBaseSocket>` is the `shortName` of the `SwCluCComProxyBaseSocket`.

8.3.3.4 Dcm Proxy

8.3.3.4.1 SwCluC_DcmProxy_Init

[SWS_SwCluC_12200] Definition of API function SwCluC_DcmProxy_Init_<SwCluCDcmBaseSocket>

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_DcmProxy_Init_<SwCluCDcmBaseSocket>
Syntax	`void SwCluC_DcmProxy_Init_<SwCluCDcmBaseSocket> ( void )`
Service ID [hex]	0x34
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_DcmProxy_Init is intended to allocate and initialize system resources used by the Dcm Proxy linked to this specific socked.
Available via	SwCluC.h

⌋

The name part `<SwCluCDcmProxyBaseSocket>` is the `shortName` of the `SwCluCDcmProxyBaseSocket`.

8.3.3.5 Dem Proxy

8.3.3.5.1 SwCluC_DemProxy_Init

[SWS_SwCluC_12300] Definition of API function SwCluC_DemProxy_Init_<SwCluCDemBaseSocket>

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_DemProxy_Init_<SwCluCDemBaseSocket>
Syntax	`void SwCluC_DemProxy_Init_<SwCluCDemBaseSocket> ( void )`
Service ID [hex]	0x35
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None

▽

△

Parameters (out)	None
Return value	None
Description	SwCluC_DemProxy_Init is intended to allocate and initialize system resources used by the Dem Proxy linked to this specific socked.
Available via	SwCluC.h

⌋

The name part <SwCluCDemProxyBaseSocket> is the shortName of the SwCluCDemProxyBaseSocket.

8.3.3.6 FiM Proxy

8.3.3.6.1 SwCluC_FiMProxy_Init

[SWS_SwCluC_12400] Definition of API function SwCluC_FiMProxy_Init_<SwCluCFiMBaseSocket>

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00310

⌈

Service Name	SwCluC_FiMProxy_Init_<SwCluCFiMBaseSocket>
Syntax	`void SwCluC_FiMProxy_Init_<SwCluCFiMBaseSocket> ( void )`
Service ID [hex]	0x36
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_FiMProxy_Init is intended to allocate and initialize system resources used by the FiM Proxy linked to this specific socked.
Available via	SwCluC.h

⌋

The name part <SwCluCFiMProxyBaseSocket> is the shortName of the SwCluCFiMProxyBaseSocket.

8.3.4 Common Functions

SwCluC does not currently define any APIs for common functions.

8.4 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.4.1 Binary Manifest

The Binary Manifest has no scheduled functions.

8.4.2 Cross Cluster Communication Scheduled functions

8.4.2.1 SwCluC_Xcc_MainFunction

[SWS_SwCluC_91002] Definition of scheduled function SwCluC_Xcc_MainFunction_<suffix>

Upstream requirements: SRS_BSW_00373, SRS_BSW_00172

⌈

Service Name	SwCluC_Xcc_MainFunction_<suffix>
Syntax	`void SwCluC_Xcc_MainFunction_<suffix> ( void )`
Service ID [hex]	0x46
Description	This function shall perform timeout monitoring of SwCluC Xcc
Available via	SwCluC.h

⌋

[SWS_SwCluC_03099]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103

⌈The Cross Cluster Communication of the Software Cluster Connection shall provide a SwCluC_Xcc_MainFunction per SwCluCXccMainFunction container where suffix is set to the shortName of the SwCluCXccMainFunction container.⌋

Please note, beside the standardized scheduled functions, the Cross Cluster Communications is free to implement additional vendor specifc ones if needed.

8.4.3 Proxy Modules Scheduled functions

The Proxy Modules have no standardized scheduled functions but are free to implement some if functionally needed.

8.5 Expected interfaces

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

8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

8.5.1.1 Binary Manifest

The `Binary Manifest` has no mandatory interfaces.

8.5.1.2 Cross Cluster Communication

The `Cross Cluster Communication` has no mandatory interfaces.

8.5.1.3 Proxy Modules

The `Proxy Modules` partly implement the mandatory interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

8.5.2.1 Binary Manifest

The `Binary Manifest` has no optional interfaces.

8.5.2.2 Cross Cluster Communication

The `Cross Cluster Communication` has no optional interfaces.

8.5.2.3 Proxy Modules

The `Proxy Modules` partly implement the optional interfaces of the according AUTOSAR BSW modules.

[SWS_SwCluC_91012] Definition of optional interfaces requested by module Sw CluC ⌈

API Function	*Header File*	*Description*
Dem_ClearPrestoredFreezeFrame	Dem.h	Clears a prestored freeze frame of a specific event. This API can only be used through the RTE and therefore no declaration is exported via Dem.h. API Availability: This API will be available only if ({ecuc(Dem/DemConfigSet/DemEvent Parameter.DemFFPrestorageSupported)} == true)
Dem_GetDebouncingOfEvent	Dem.h	Gets the debouncing status of an event. This function shall not be used for EventId with native debouncing within their functions. It is rather for EventIds using debouncing within the Dem.
Dem_GetDTCOfEvent	Dem.h	Gets the DTC of an event.
Dem_GetEventAvailable	Dem.h	Get the Event's availability.
Dem_GetEventExtendedDataRecordEx	Dem.h	Gets the data of an extended data record by event.
Dem_GetEventFreezeFrameDataEx	Dem.h	Gets the data of a freeze frame by event.
Dem_GetEventUdsStatus	Dem.h	Gets the current UDS status byte assigned to the DTC for the event
Dem_GetFaultDetectionCounter	Dem.h	Gets the fault detection counter of an event. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.
Dem_GetMonitorStatus	Dem.h	Gets the current monitor status for an event.
Dem_PrestoreFreezeFrame	Dem.h	Captures the freeze frame data for a specific event. This API can only be used through the RTE and therefore no declaration is exported via Dem.h. API Availability: This API will be available only if ({ecuc(Dem/DemConfigSet/DemEvent Parameter.DemFFPrestorageSupported)} == true)
Dem_ResetEventDebounceStatus	Dem.h	Control the internal debounce counter/timer by BSW modules and SW-Cs. The event qualification will not be affected by these debounce state changes. This API is available for BSW modules as soon as Dem_PreInit has been completed (refer to [SWS_Dem_00438] and [SWS_Dem_00167]).
Dem_ResetEventStatus	Dem.h	Resets the event failed status. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.
Dem_SetEventDisabled	Dem.h	Service for reporting the event as disabled to the Dem for the PID $41 computation. API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Dem_SetEventStatusWithMonitorData	Dem.h	This API will be available only if ({Dem/DemConfig Set/DemEventParameter/DemEventReportingType} == STANDARD_REPORTING_WITH_MONITOR_DATA)

▽

△

API Function	Header File	Description
Dem_SetWIRStatus	Dem.h	Sets the WIR status bit via failsafe SW-Cs. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.
NvM_CancelJobs	NvM.h	Service to cancel all jobs pending for a NV block.
NvM_EraseNvBlock	NvM.h	Service to erase a NV block.
NvM_GetDataIndex	NvM.h	Service for getting the currently set DataIndex of a dataset NVRAM block
NvM_GetErrorStatus	NvM.h	Service to read the block dependent error/status information.
NvM_InvalidateNvBlock	NvM.h	Service to invalidate a NV block.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_ReadPRAMBlock	NvM.h	Service to copy the data of the NV block to its corresponding permanent RAM block.
NvM_RestoreBlockDefaults	NvM.h	Service to restore the default data to its corresponding RAM block.
NvM_RestorePRAMBlockDefaults	NvM.h	Service to restore the default data to its corresponding permanent RAM block.
NvM_SetBlockProtection	NvM.h	Service for setting/resetting the write protection for a NV block.
NvM_SetDataIndex	NvM.h	Service for setting the DataIndex of a dataset NVRAM block.
NvM_SetRamBlockStatus	NvM.h	Service for setting the RAM block status of a permanent RAM block or the status of the explicit synchronization of a NVRAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.
NvM_WritePRAMBlock	NvM.h	Service to copy the data of the permanent RAM block to its corresponding NV block.

⌋

The full list of used interfaces is documented in appendix D.

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

8.5.3.1 Binary Manifest

The `Binary Manifest` has no configurable interfaces.

8.5.3.2 Cross Cluster Communication

The `Cross Cluster Communication` implements the interfaces of a `Cross Software Cluster Communication Plug-In` as specified in document [4].

8.5.3.3 Proxy Modules

The `Proxy Modules` partly implement the configurable interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.5.4 Callout interfaces

No Content.

8.6 Service Interfaces

8.6.1 Binary Manifest

The `Binary Manifest` has no Service Interfaces.

8.6.2 Cross Cluster Communication

The `Cross Cluster Communication` has no Service Interfaces.

8.6.3 Proxy Modules

The `Proxy Modules` only implement Service Interfaces, which are already defined by the original AUTOSAR Service. Those are listed in appendix E.

9 Sequence diagrams

No sequence diagrams are contained in this section.

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 `SwCluC`.

Chapter 10.3 specifies published information of the module `SwCluC`.

10.1 How to read this chapter

For details refer to [3] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 Module Configuration

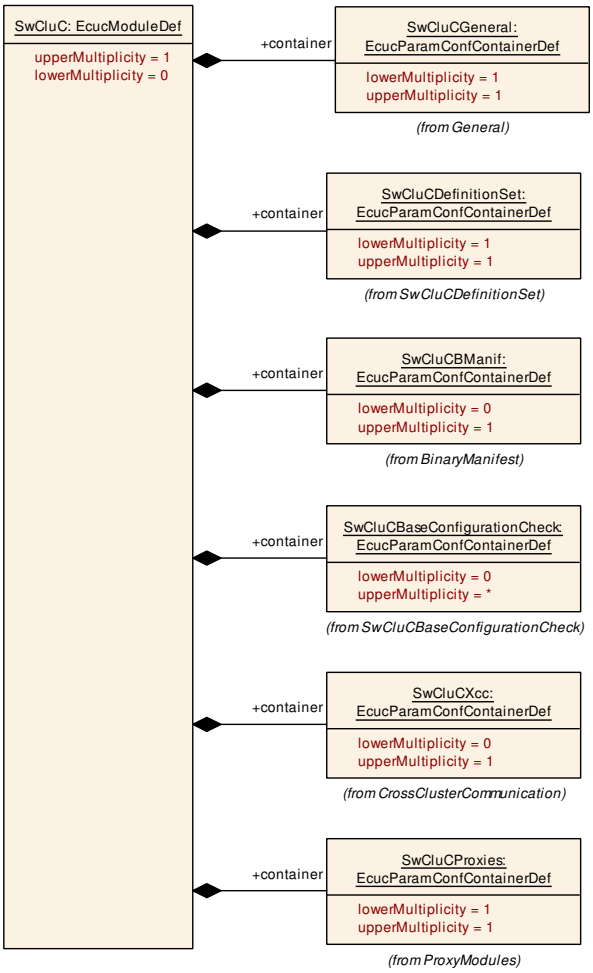

Figure 10.1: SwCluC configuration overview

[ECUC_SwCluC_00001] Definition of EcucModuleDef SwCluC ⌈

Module Name	SwCluC
Description	Module to collect Software Cluster Connection specific configuration information.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCBManif	0..1	Configuration of the Binary Manifest of the Software Cluster Connection
SwCluCBaseConfigurationCheck	0..*	Configuration of the base configuration check of Software Cluster Connection. This configuration places a mandatory entry in the Binary Manifest where the guard value is used to proof if the general setup of the Application Software Cluster is compatible to the general setup of the Host Software Cluster. For each Application Software Cluster an individual base configuration check should be defined.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCDefinitionSet	1	Definition of a set of Software Clusters
SwCluCGeneral	1	General configuration of the Software Cluster Connection
SwCluCProxies	1	General configuration of the Proxy Modules of Software Cluster Connection
SwCluCXcc	0..1	Configuration of the Binary Manifest of the Software Cluster Connection

⌋

10.2.2 General configuration parameters

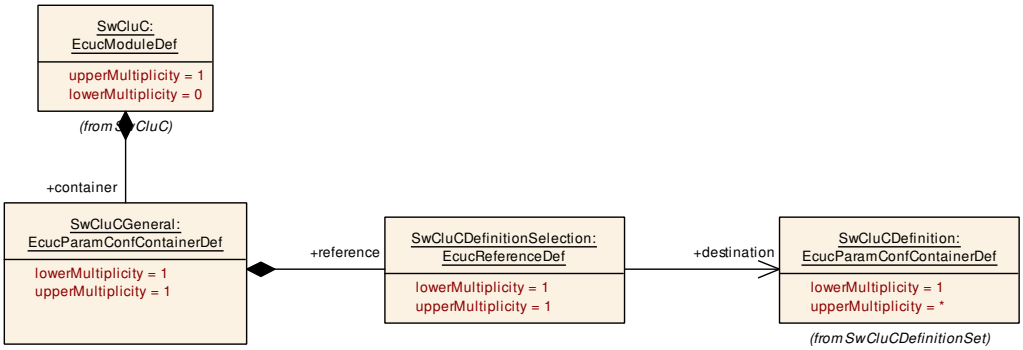

Figure 10.2: SwCluC General configuration parameters

[ECUC_SwCluC_00002] Definition of EcucParamConfContainerDef SwCluCGeneral ⌈

Container Name	SwCluCGeneral
Parent Container	SwCluC
Upstream mapping defined	–
Description	General configuration of the Software Cluster Connection
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCDefinitionSelection	1	[ECUC_SwCluC_00005]

No Included Containers

⌋

[ECUC_SwCluC_00005] Definition of EcucReferenceDef SwCluCDefinitionSelection ⌈

Parameter Name	SwCluCDefinitionSelection
Parent Container	SwCluCGeneral
Upstream mapping defined	–
Description	This reference selects the Software Cluster Definition which is applied for this Software Cluster.
Multiplicity	1
Type	Reference to SwCluCDefinition
Pre-Compile only	true

⌋

10.2.3 Software Cluster Definition

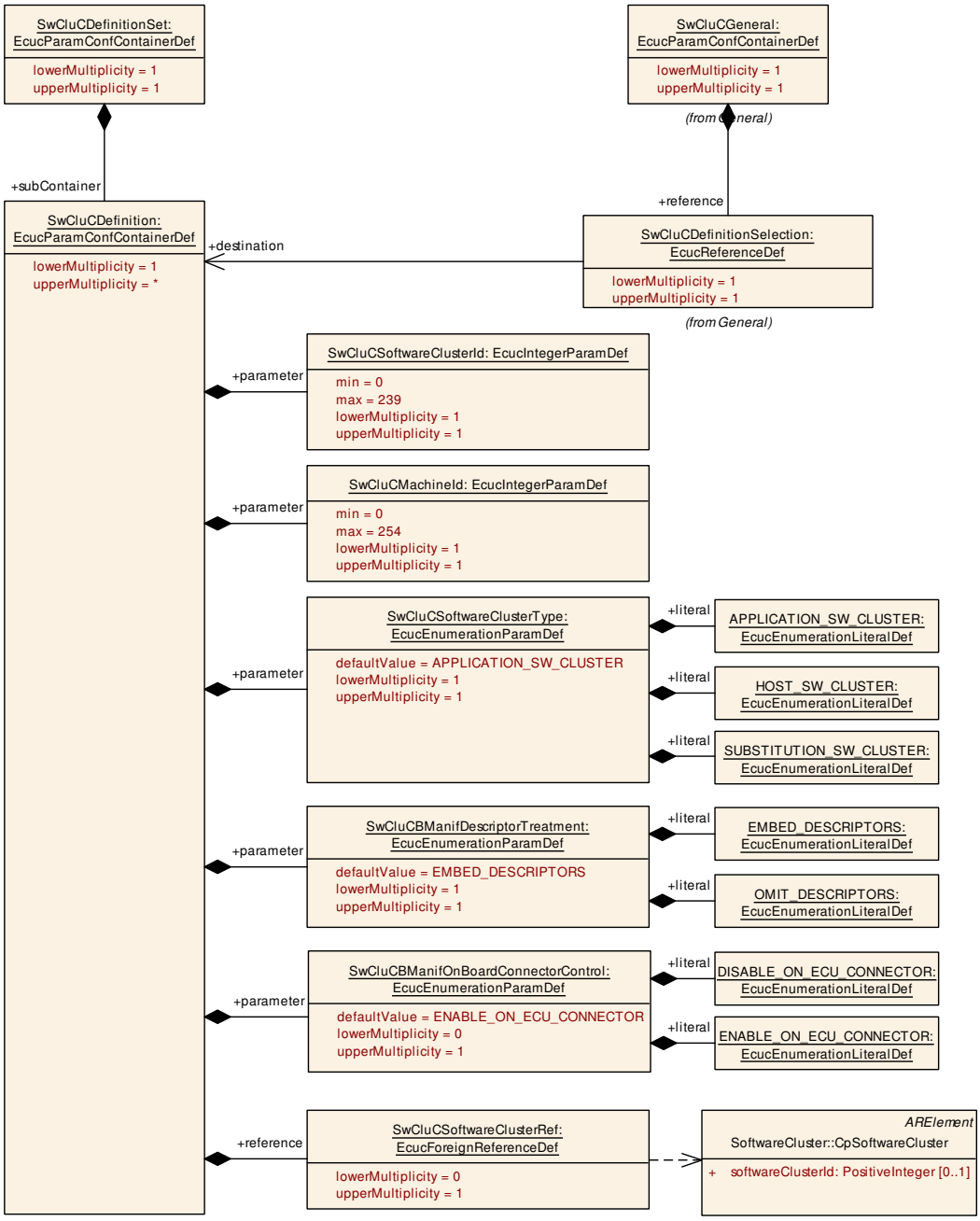

Figure 10.3: SwCluC Software Cluster Definition

[ECUC_SwCluC_00004] Definition of EcucParamConfContainerDef SwCluCDefinitionSet ⌈

Container Name	SwCluCDefinitionSet
Parent Container	SwCluC
Upstream mapping defined	–

▽

△

Description	Definition of a set of Software Clusters
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
SwCluCDefinition	1..*	Definition of the general properties of the Software Cluster. The defintions are separated in a set of defintions in order to enable an holistic view about all exsiting Software Cluster Definitions.

⌋

[ECUC_SwCluC_00006] Definition of EcucParamConfContainerDef SwCluCDefinition ⌈

Container Name	SwCluCDefinition
Parent Container	SwCluCDefinitionSet
Upstream mapping defined	–
Description	Definition of the general properties of the Software Cluster. The defintions are separated in a set of defintions in order to enable an holistic view about all exsiting Software Cluster Definitions.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCBManifDescriptorTreatment	1	[ECUC_SwCluC_00010]
SwCluCBManifOnBoardConnectorControl	0..1	[ECUC_SwCluC_00011]
SwCluCMachineId	1	[ECUC_SwCluC_00008]
SwCluCSoftwareClusterId	1	[ECUC_SwCluC_00007]
SwCluCSoftwareClusterType	1	[ECUC_SwCluC_00009]
SwCluCSoftwareClusterRef	0..1	[ECUC_SwCluC_00034]

No Included Containers

⌋

[ECUC_SwCluC_00010] Definition of EcucEnumerationParamDef SwCluCBManifDescriptorTreatment ⌈

Parameter Name	SwCluCBManifDescriptorTreatment
Parent Container	SwCluCDefinition
Upstream mapping defined	–

▽

△

Description	Configures the existence of the Interface Descriptor Table in the Binary Object of the Software Cluster.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	EMBED_DESCRIPTORS	The Interface Descriptor Table of the Binary Manifest is embed into the Binary Object.
	OMIT_DESCRIPTORS	The Interface Descriptor Table of the Binary Manifest is omitted from the Binary Object. The information needs to be delivered as CpSoftware ClusterBinaryManifestDescriptor.
Default value	EMBED_DESCRIPTORS	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00011] Definition of EcucEnumerationParamDef SwCluCBManifOnBoardConnectorControl ⌈

Parameter Name	SwCluCBManifOnBoardConnectorControl	
Parent Container	SwCluCDefinition	
Upstream mapping defined	–	
Description	SwCluCBManifOnBoardConnectorControl enables or disables the onECU execution of the Software Cluster Connector. This setting is only applicable for the Host Software Cluster.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	DISABLE_ON_ECU_CONNECTOR	The connection of Software Clusters OnECU is disabled for this machine.
	ENABLE_ON_ECU_CONNECTOR	The connection of Software Clusters OnECU is enabled for this machine.
Default value	ENABLE_ON_ECU_CONNECTOR	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00008] Definition of EcucIntegerParamDef SwCluCMachineId ⌈

Parameter Name	SwCluCMachineId	
Parent Container	SwCluCDefinition	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Unique number of the (virtual or physical) machine to which the Software Cluster belongs.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 254	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00007] Definition of EcucIntegerParamDef SwCluCSoftwareClusterId ⌈

Parameter Name	SwCluCSoftwareClusterId	
Parent Container	SwCluCDefinition	
Upstream mapping defined	–	
Description	Unique number of the Software Cluster Numbers >= 0xFE are reserved to indicate special values or the unconnected state.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 239	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00009] Definition of EcucEnumerationParamDef SwCluCSoftwareClusterType ⌈

Parameter Name	SwCluCSoftwareClusterType	
Parent Container	SwCluCDefinition	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The type of the Software Cluster	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	APPLICATION_SW_CLUSTER	This Software Cluster is an Application Software Cluster.
	HOST_SW_CLUSTER	This Software Cluster is a Host Software Cluster.
	SUBSTITUTION_SW_CLUSTER	This Software Cluster is a Substitution Software Cluster.
Default value	APPLICATION_SW_CLUSTER	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00034] Definition of EcucForeignReferenceDef SwCluCSoftwareClusterRef ⌈

Parameter Name	SwCluCSoftwareClusterRef
Parent Container	SwCluCDefinition
Upstream mapping defined	–
Description	Reference to the CpSoftwareCluster to define the link to the Software Cluster description in the System.
Multiplicity	0..1
Type	Foreign reference to CP-SOFTWARE-CLUSTER
Pre-Compile only	true

⌋

[SWS_SwCluC_CONSTR_00096] `SwCluCSoftwareClusterId` shall be set accoring `CpSoftwareCluster.softwareClusterId`

Upstream requirements: SRS_SwCluC_00212, SRS_BSW_00167

⌈For a specific `SwCluCDefinition`, the value of `SwCluCDefinition.SwCluCSoftwareClusterId` shall be identical to the value of the `softwareClusterId` of the `CpSoftwareCluster` referenced via `SwCluCDefinition.SwCluCSoftwareClusterRef`.⌋

Rationale: The `SwCluCSoftwareClusterId` cannot be chosen by the ECU integrator, if the `softwareClusterId` is already given by a system description. For integration specific software clusters, the ECU integrator has to choose a free number for the `SwCluCSoftwareClusterId`.

10.2.4 Software Cluster Base Configuration Check

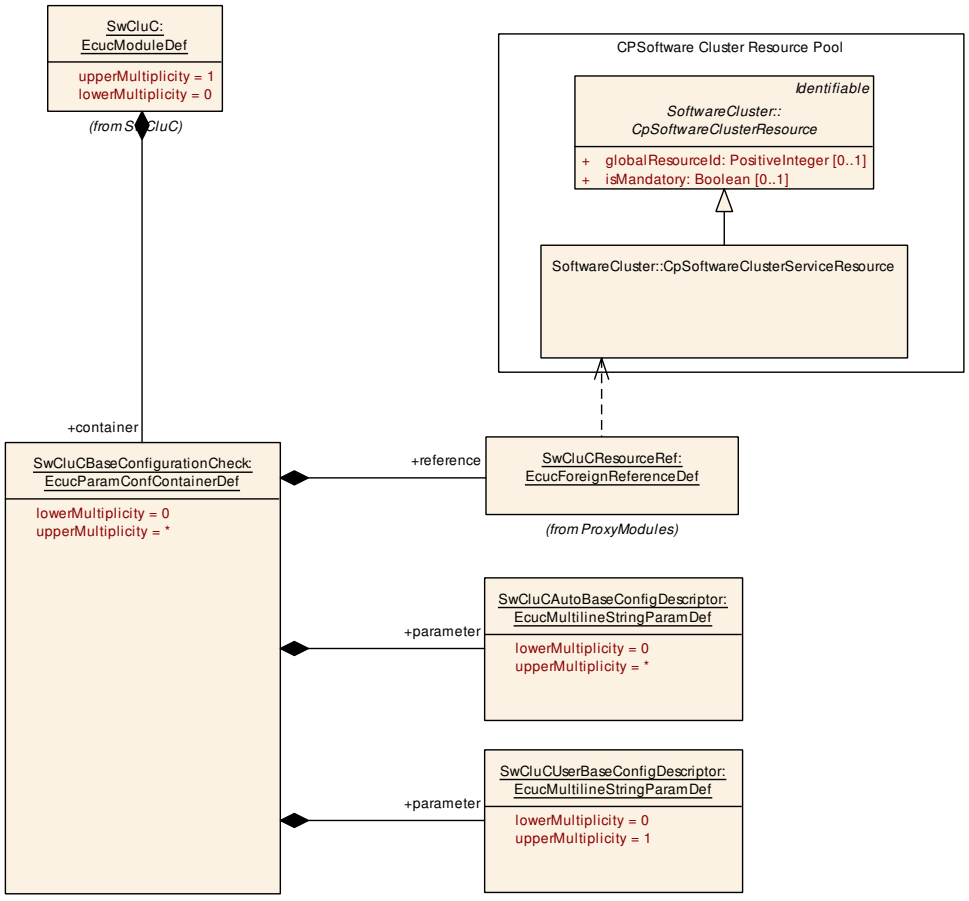

Figure 10.4: Software Cluster Base Configuration Check

[ECUC_SwCluC_00079] Definition of EcucParamConfContainerDef SwCluCBaseConfigurationCheck ⌈

Container Name	SwCluCBaseConfigurationCheck
Parent Container	SwCluC
Upstream mapping defined	–
Description	Configuration of the base configuration check of Software Cluster Connection. This configuration places a mandatory entry in the Binary Manifest where the guard value is used to proof if the general setup of the Application Software Cluster is compatible to the general setup of the Host Software Cluster. For each Application Software Cluster an individual base configuration check should be defined.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCAutoBaseConfigDescriptor	0..*	[ECUC_SwCluC_00080]
SwCluCUserBaseConfigDescriptor	0..1	[ECUC_SwCluC_00081]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00080] Definition of EcucMultilineStringParamDef SwCluCAutoBaseConfigDescriptor ⌈

Parameter Name	SwCluCAutoBaseConfigDescriptor
Parent Container	SwCluCBaseConfigurationCheck
Upstream mapping defined	–
Description	A machine determined string which represent the basic configuration assumption. Multiple SwCluCAutoBaseConfigDescriptor values can be added to support that different tools contribute to the checksum. For instance Software Cluster Connection Tool, Compiler, Link, or Locate tooling. isAuto shall be set to true since tool chain adds its collected values automatically.
Multiplicity	0..*
Type	EcucMultilineStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00081] Definition of EcucMultilineStringParamDef SwCluCUserBaseConfigDescriptor ⌈

Parameter Name	SwCluCUserBaseConfigDescriptor
Parent Container	SwCluCBaseConfigurationCheck
Upstream mapping defined	–
Description	A user determined string which represent the basic configuration assumption.
Multiplicity	0..1
Type	EcucMultilineStringParamDef

▽

△

Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00087] Definition of EcucForeignReferenceDef SwCluCResourceRef ⌈

Parameter Name	SwCluCResourceRef
Parent Container	SwCluCBaseConfigurationCheck, SwCluCComProxyBaseSocket, SwCluCDcmProxyBaseSocket, SwCluCDcmProxyDiagnosticData, SwCluCDcmProxyRoutineControl, SwCluCDcmProxyServiceRequestNotification, SwCluCDemProxyBaseSocket, SwCluCDemProxyDiagnosticEvent, SwCluCFiMProxyBaseSocket, SwCluCFiMProxyFID, SwCluCNvMBaseSocket, SwCluCNvMConfigIDProxyNvBlock, SwCluCNvMProxyNvBlock, SwCluCOsProxyFunctionDispatcher, SwCluCOsProxyOsApplication, SwCluCOsProxyOsBaseSocket, SwCluCOsProxyOsResource, SwCluCOsProxyOsSpinlock, SwCluCOsProxyOsTask, SwCluCXccBaseSocket
Upstream mapping defined	–
Description	Reference to the CpSoftwareClusterServiceResource.
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Pre-Compile only	true

⌋

[SWS_SwCluC_CONSTR_00078] `SwCluCBaseConfigurationCheck` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_BASE_CNF`

Upstream requirements: SRS_SwCluC_00212, SRS_BSW_00167

⌈The `SwCluCBaseConfigurationCheck.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_BASE_CNF`.⌋

10.2.5 Binary Manifest

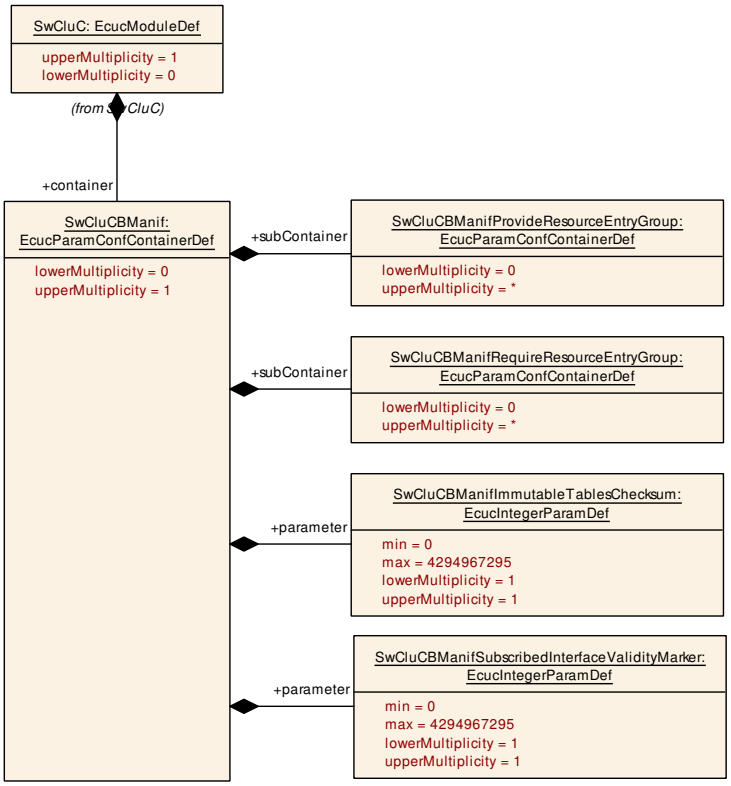

Figure 10.5: SwCluC Binary Manifest

[ECUC_SwCluC_00003] Definition of EcucParamConfContainerDef SwCluCBManif ⌈

Container Name	SwCluCBManif
Parent Container	SwCluC
Upstream mapping defined	–
Description	Configuration of the Binary Manifest of the Software Cluster Connection
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifImmutableTablesChecksum	1	[ECUC_SwCluC_00020]
SwCluCBManifSubscribedInterfaceValidityMarker	1	[ECUC_SwCluC_00022]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCBManifProvideResource EntryGroup	0..*	Describes a group of provided resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCBManifRequireResourceEntryGroup	0..*	Describes a group of required resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access.
SwCluCBManifResourceType	0..*	This container defines the structure of a resource type in the Binary Manifest.

⌋

[ECUC_SwCluC_00020] Definition of EcucIntegerParamDef SwCluCBManifImmutableTablesChecksum ⌈

Parameter Name	SwCluCBManifImmutableTablesChecksum	
Parent Container	SwCluCBManif	
Upstream mapping defined	–	
Description	This parameter defines the initialization value of the Immutable Tables Checksum. If the Binary Manifest code generator already calculates the checksum, the value can be stored here (withAuto == true).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00022] Definition of EcucIntegerParamDef SwCluCBManifSubscribedInterfaceValidityMarker ⌈

Parameter Name	SwCluCBManifSubscribedInterfaceValidityMarker	
Parent Container	SwCluCBManif	
Upstream mapping defined	–	
Description	This parameter defines the initialization value of the Subscribed Interface Validity Marker. The init value should be set to the erased value of the flash memory.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

10.2.5.1 Provide Resource Entry Group

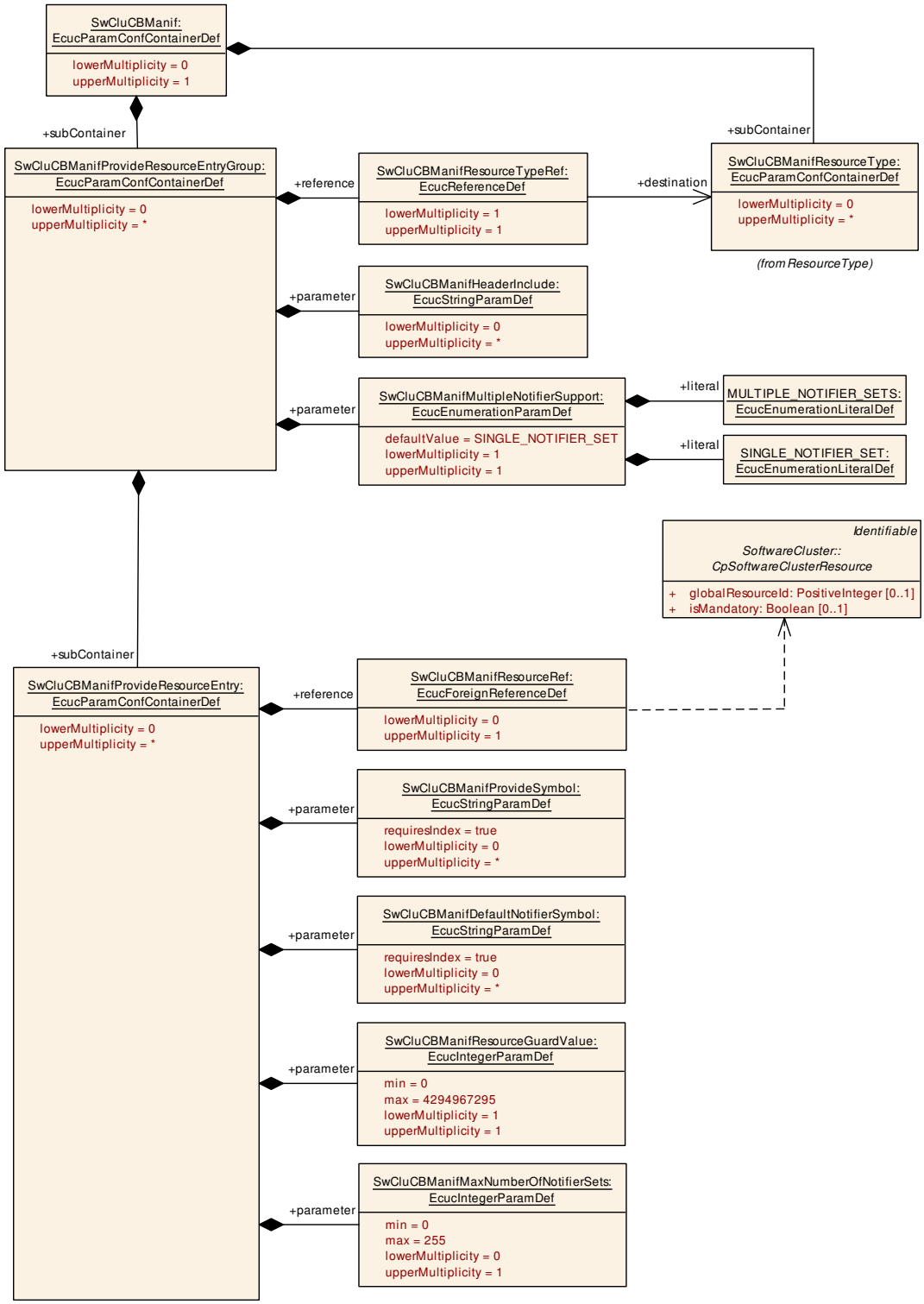

Figure 10.6: SwCluC Provide Resource Entry Group

[ECUC_SwCluC_00018] Definition of EcucParamConfContainerDef SwCluCB-ManifProvideResourceEntryGroup ⌈

Container Name	SwCluCBManifProvideResourceEntryGroup
Parent Container	SwCluCBManif
Upstream mapping defined	–
Description	Describes a group of provided resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifHeaderInclude	0..*	[ECUC_SwCluC_00024]
SwCluCBManifMultipleNotifierSupport	1	[ECUC_SwCluC_00016]
SwCluCBManifResourceTypeRef	1	[ECUC_SwCluC_00023]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCBManifProvideResource Entry	0..*	Describes a single entry for a provided resource in the Binary Manifest.

⌋

[ECUC_SwCluC_00024] Definition of EcucStringParamDef SwCluCBManifHeader Include ⌈

Parameter Name	SwCluCBManifHeaderInclude
Parent Container	SwCluCBManifProvideResourceEntryGroup, SwCluCBManifRequireResourceEntry Group
Upstream mapping defined	–
Description	Defines the header file(s) where the owner of the SwCluCBManifProvideResource EntryGroup / SwCluCBManifRequireResourceEntryGroup has the declarations of the symbols for the handle initialization.
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00016] Definition of EcucEnumerationParamDef SwCluCBManif MultipleNotifierSupport ⌈

Parameter Name	SwCluCBManifMultipleNotifierSupport
Parent Container	SwCluCBManifProvideResourceEntryGroup
Upstream mapping defined	–
Description	SwCluCBManifMultipleNotifierSupport defines whether multiple Software Cluster can register on the notifier handles. For each possible connection a set of notifier handles is allocated.

▽

△

Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	MULTIPLE_NOTIFIER_SETS	The provided resource supports the co-resident notifier connections to multiple required resources.
	SINGLE_NOTIFIER_SET	The provided resource supports at most one connection of notifiers to a required resource.
Default value	SINGLE_NOTIFIER_SET	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00023] Definition of EcucReferenceDef SwCluCBManifResourceTypeRef ⌈

Parameter Name	SwCluCBManifResourceTypeRef
Parent Container	SwCluCBManifProvideResourceEntryGroup, SwCluCBManifRequireResourceEntryGroup
Upstream mapping defined	–
Description	Defines the resource type for all SwCluCBManifProvideResourceEntry / SwCluCBManifRequireResourceEntry
Multiplicity	1
Type	Reference to SwCluCBManifResourceType
Pre-Compile only	true

⌋

10.2.5.2 Provide Resource Entry

[ECUC_SwCluC_00164] Definition of EcucParamConfContainerDef SwCluCBManifProvideResourceEntry ⌈

Container Name	SwCluCBManifProvideResourceEntry
Parent Container	SwCluCBManifProvideResourceEntryGroup
Upstream mapping defined	–
Description	Describes a single entry for a provided resource in the Binary Manifest.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifDefaultNotifierSymbol	0..*	[ECUC_SwCluC_00027]
SwCluCBManifMaxNumberOfNotifierSets	0..1	[ECUC_SwCluC_00029]
SwCluCBManifProvideSymbol	0..*	[ECUC_SwCluC_00026]
SwCluCBManifResourceGuardValue	1	[ECUC_SwCluC_00028]
SwCluCBManifResourceRef	0..1	[ECUC_SwCluC_00025]

No Included Containers

⌋

[ECUC_SwCluC_00027] Definition of EcucStringParamDef SwCluCBManifDefaultNotifierSymbol ⌈

Parameter Name	SwCluCBManifDefaultNotifierSymbol
Parent Container	SwCluCBManifProvideResourceEntry
Upstream mapping defined	–
Description	SwCluCBManifDefaultNotifierSymbol set the default value of a Notifier Handle put into the Modifiable Interface. The number and order of SwCluCBManifDefaultNotifier Symbol needs to match the number and order of SwCluCBManifNotifierHandle in the referenced SwCluCBManifResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001 CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00029] Definition of EcucIntegerParamDef SwCluCBManifMaxNumberOfNotifierSets ⌈

Parameter Name	SwCluCBManifMaxNumberOfNotifierSets	
Parent Container	SwCluCBManifProvideResourceEntry	
Upstream mapping defined	–	
Description	SwCluCBManifMaxNumberOfNotifierSets defines the maximum number of possible required resources using notifier handles. It is required in case SwCluCBManifMultiple NotifierSupport is set to MULTIPLE_NOTIFIER_SETS.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00026] Definition of EcucStringParamDef SwCluCBManifProvideSymbol ⌈

Parameter Name	SwCluCBManifProvideSymbol
Parent Container	SwCluCBManifProvideResourceEntry
Upstream mapping defined	–

▽

△

Description	SwCluCBManifProvideSymbol set the value of a Provide Handle put into the Immutable Interface. The number and order of SwCluCBManifProvideSymbols needs to match the number and order of SwCluCBManifProvideHandle in the referenced SwCluCBManif ResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001 CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00028] Definition of EcucIntegerParamDef SwCluCBManifResourceGuardValue ⌈

Parameter Name	SwCluCBManifResourceGuardValue
Parent Container	SwCluCBManifProvideResourceEntry, SwCluCBManifRequireResourceEntry
Upstream mapping defined	–
Description	SwCluCBManifResourceGuardValue provides guarding information checked by the Software Cluster connector. A provided resources and required resources of Software Clusters can only be connected, if the resource guarding values of provider and requester are equal.
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00025] Definition of EcucForeignReferenceDef SwCluCBManifResourceRef ⌈

Parameter Name	SwCluCBManifResourceRef
Parent Container	SwCluCBManifProvideResourceEntry, SwCluCBManifRequireResourceEntry
Upstream mapping defined	–
Description	Reference to the CpSoftwareClusterResource determining the global resource Id. If the reference is not set, the global resource Id in the Binary Manifest will be set to 0 for this resource entry in order to indicate an invalid resource.
Multiplicity	0..1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-RESOURCE
Pre-Compile only	true

⌋

10.2.5.3 Require Resource Entry Group

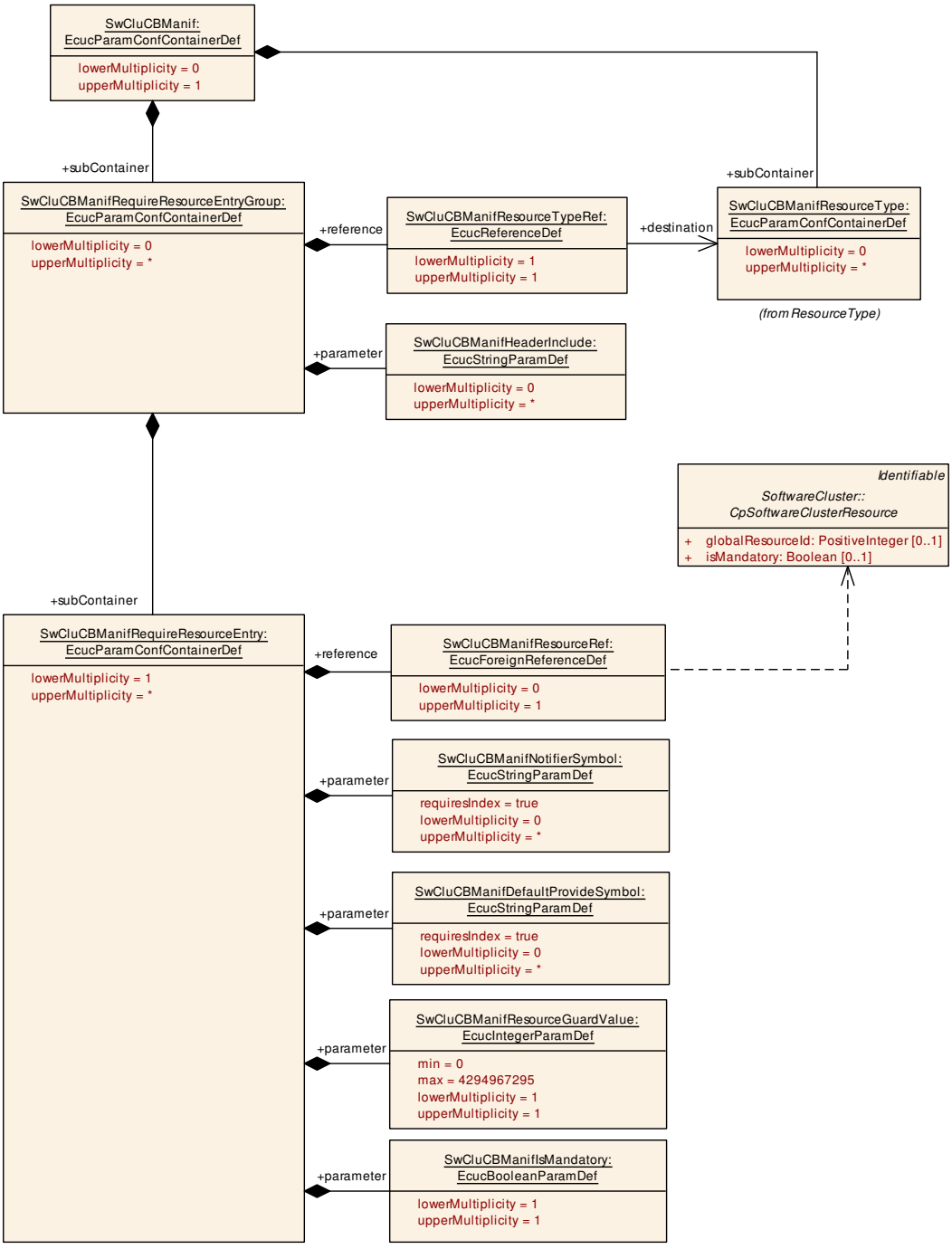

Figure 10.7: SwCluC Require Resource Entry Group

[ECUC_SwCluC_00019] Definition of EcucParamConfContainerDef SwCluCB-ManifRequireResourceEntryGroup ⌈

Container Name	SwCluCBManifRequireResourceEntryGroup
Parent Container	SwCluCBManif
Upstream mapping defined	–
Description	Describes a group of required resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCBManifHeaderInclude	0..*	[ECUC_SwCluC_00024]
SwCluCBManifResourceTypeRef	1	[ECUC_SwCluC_00023]

Included Containers		
Container Name	Multiplicity	Description
SwCluCBManifRequireResource Entry	1..*	Describes a single entry for a required resource in the Binary Manifest.

⌋

For parameter table [ECUC_SwCluC_00024] SwCluCBManifHeaderInclude, see definition below container SwCluCBManifProvideResourceEntryGroup.

For parameter table [ECUC_SwCluC_00023] SwCluCBManifResourceTypeRef, see definition below container SwCluCBManifProvideResourceEntryGroup.

10.2.5.4 Require Resource Entry

[ECUC_SwCluC_00165] Definition of EcucParamConfContainerDef SwCluCBManifRequireResourceEntry ⌈

Container Name	SwCluCBManifRequireResourceEntry
Parent Container	SwCluCBManifRequireResourceEntryGroup
Upstream mapping defined	–
Description	Describes a single entry for a required resource in the Binary Manifest.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCBManifDefaultProvideSymbol	0..*	[ECUC_SwCluC_00031]
SwCluCBManifIsMandatory	1	[ECUC_SwCluC_00032]
SwCluCBManifNotifierSymbol	0..*	[ECUC_SwCluC_00030]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifResourceGuardValue	1	[ECUC_SwCluC_00028]
SwCluCBManifResourceRef	0..1	[ECUC_SwCluC_00025]

No Included Containers

⌋

[ECUC_SwCluC_00031] Definition of EcucStringParamDef SwCluCBManifDefaultProvideSymbol ⌈

Parameter Name	SwCluCBManifDefaultProvideSymbol
Parent Container	SwCluCBManifRequireResourceEntry
Upstream mapping defined	–
Description	SwCluCBManifDefaultProvideSymbol set the default value of a Provide Handle put into the Modifiable Interface. The number and order of SwCluCBManifDefaultProvide Symbols needs to match the number and order of SwclCBManifProvideHandles in the referenced SwCluCBManifResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001 CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00032] Definition of EcucBooleanParamDef SwCluCBManifIsMandatory ⌈

Parameter Name	SwCluCBManifIsMandatory
Parent Container	SwCluCBManifRequireResourceEntry
Upstream mapping defined	–
Description	SwCluCBManifIsMandatoryindicates, that the resource is mandatory to operate this Software Cluster. If the resource is not provided on the machine the connection process for this Software Cluster requiring this resource gets aborted.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00030] Definition of EcucStringParamDef SwCluCBManifNotifierSymbol ⌈

Parameter Name	SwCluCBManifNotifierSymbol
Parent Container	SwCluCBManifRequireResourceEntry
Upstream mapping defined	–
Description	SwCluCBManifNotifierSymbol set the value of a Notifier Handle put into the Immutable Interface. The number and order of SwCluCBManifNotifierSymbol needs to match the number and order of SwclCBManifNotifierHandles in the referenced SwCluCBManif ResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001 CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00028] SwCluCBManifResourceGuardValue, see definition below container SwCluCBManifProvideResourceEntry.

For parameter table [ECUC_SwCluC_00025] SwCluCBManifResourceRef, see definition below container SwCluCBManifProvideResourceEntry.

10.2.5.5 Resource Type

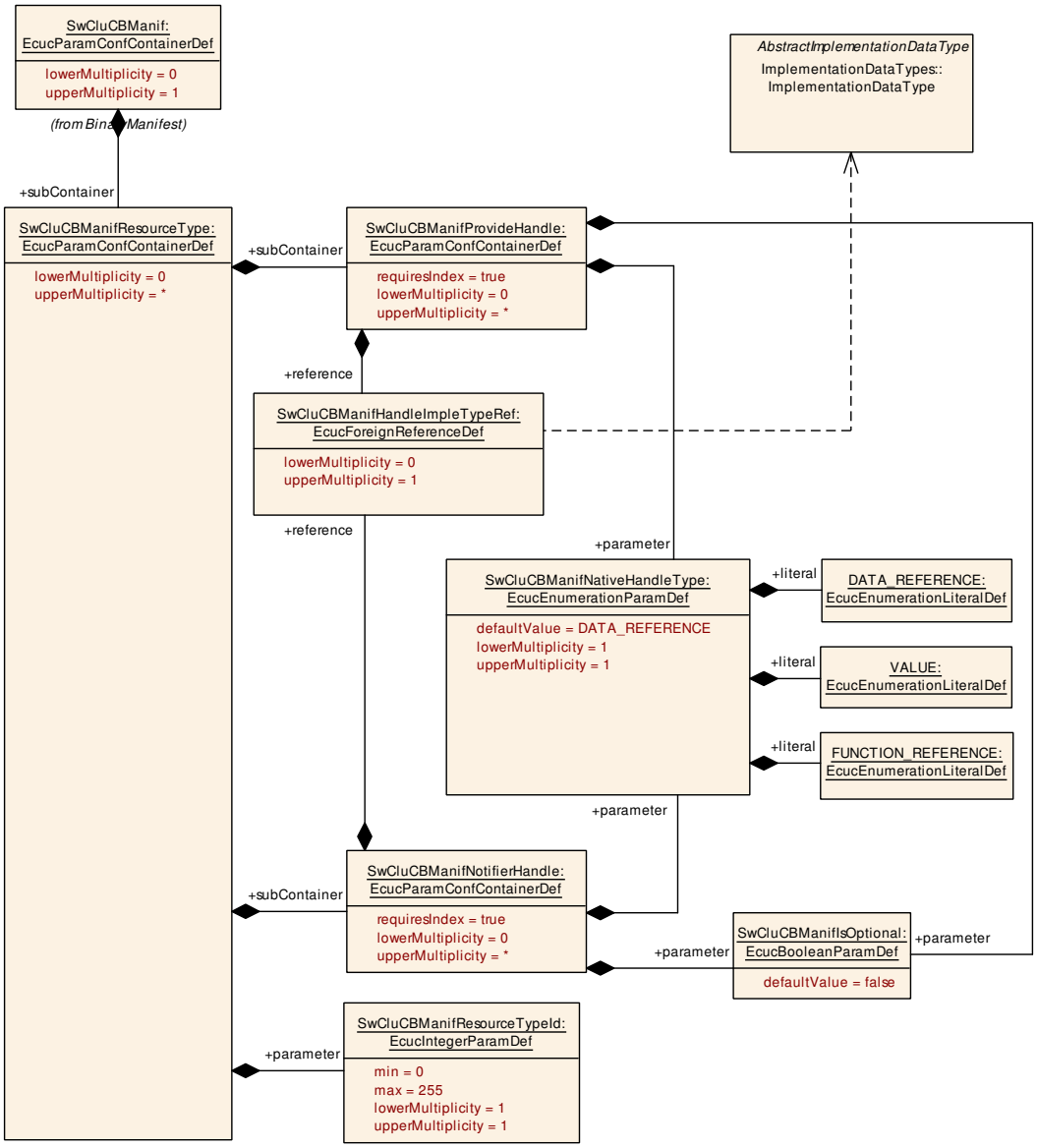

Figure 10.8: SwCluC resource type

[ECUC_SwCluC_00012] Definition of EcucParamConfContainerDef SwCluCBManifResourceType ⌈

Container Name	SwCluCBManifResourceType
Parent Container	SwCluCBManif
Upstream mapping defined	–
Description	This container defines the structure of a resource type in the Binary Manifest.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifResourceTypeId	1	[ECUC_SwCluC_00017]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCBManifNotifierHandle	0..*	Defines a Notifier Handle and its properties for this resource type.
SwCluCBManifProvideHandle	0..*	Defines a Provide Handle and its properties for this resource type.

⌋

[ECUC_SwCluC_00017] Definition of EcucIntegerParamDef SwCluCBManifResourceTypeId ⌈

Parameter Name	SwCluCBManifResourceTypeId	
Parent Container	SwCluCBManifResourceType	
Upstream mapping defined	–	
Description	Unique number of the resource type.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00013] Definition of EcucParamConfContainerDef SwCluCBManifProvideHandle ⌈

Container Name	SwCluCBManifProvideHandle
Parent Container	SwCluCBManifResourceType
Upstream mapping defined	–
Description	Defines a Provide Handle and its properties for this resource type. **Attributes:** requiresIndex=true
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifIsOptional	1	[ECUC_SwCluC_00099]
SwCluCBManifNativeHandleType	1	[ECUC_SwCluC_00033]
SwCluCBManifHandleImpleTypeRef	0..1	[ECUC_SwCluC_00015]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00099] SwCluCBManifIsOptional, see definition below container SwCluCBManifNotifierHandle.

For parameter table [ECUC_SwCluC_00033] SwCluCBManifNativeHandleType, see definition below container SwCluCBManifNotifierHandle.

For parameter table [ECUC_SwCluC_00015] SwCluCBManifHandleImpleTypeRef, see definition below container SwCluCBManifNotifierHandle.

[ECUC_SwCluC_00014] Definition of EcucParamConfContainerDef SwCluCBManifNotifierHandle ⌈

Container Name	SwCluCBManifNotifierHandle
Parent Container	SwCluCBManifResourceType
Upstream mapping defined	–
Description	Defines a Notifier Handle and its properties for this resource type. **Attributes:** requiresIndex=true
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCBManifIsOptional	1	[ECUC_SwCluC_00099]
SwCluCBManifNativeHandleType	1	[ECUC_SwCluC_00033]
SwCluCBManifHandleImpleTypeRef	0..1	[ECUC_SwCluC_00015]

No Included Containers

⌋

[ECUC_SwCluC_00099] Definition of EcucBooleanParamDef SwCluCBManifIsOptional ⌈

Parameter Name	SwCluCBManifIsOptional
Parent Container	SwCluCBManifNotifierHandle, SwCluCBManifProvideHandle
Upstream mapping defined	–
Description	Indicates that this handle is optional and may not always exist for each Provide Resource Entry or Require Resource Entry.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_SwCluC_00033] Definition of EcucEnumerationParamDef SwCluCBManifNativeHandleType ⌈

Parameter Name	SwCluCBManifNativeHandleType	
Parent Container	SwCluCBManifNotifierHandle, SwCluCBManifProvideHandle	
Upstream mapping defined	–	
Description	This paramter determines the unterlying native type of the handle which can be • pointer to variable • value • pointer to function This information is required to correctly initialize the handle in the Immutable Interface or Modifiable Interface.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DATA_REFERENCE	Handle is pointer to variable
	FUNCTION_REFERENCE	Handle is pointer to function
	VALUE	Handle is a value
Default value	DATA_REFERENCE	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00015] Definition of EcucForeignReferenceDef SwCluCBManifHandleImpleTypeRef ⌈

Parameter Name	SwCluCBManifHandleImpleTypeRef
Parent Container	SwCluCBManifNotifierHandle, SwCluCBManifProvideHandle
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]
Description	Reference to the ImplementationDataType of the handle. This type can be additionally configured to get a correct casting of the native handle type to the returned handel type.
Multiplicity	0..1
Type	Foreign reference to IMPLEMENTATION-DATA-TYPE
Pre-Compile only	true

⌋

10.2.6 Cross Cluster Communication

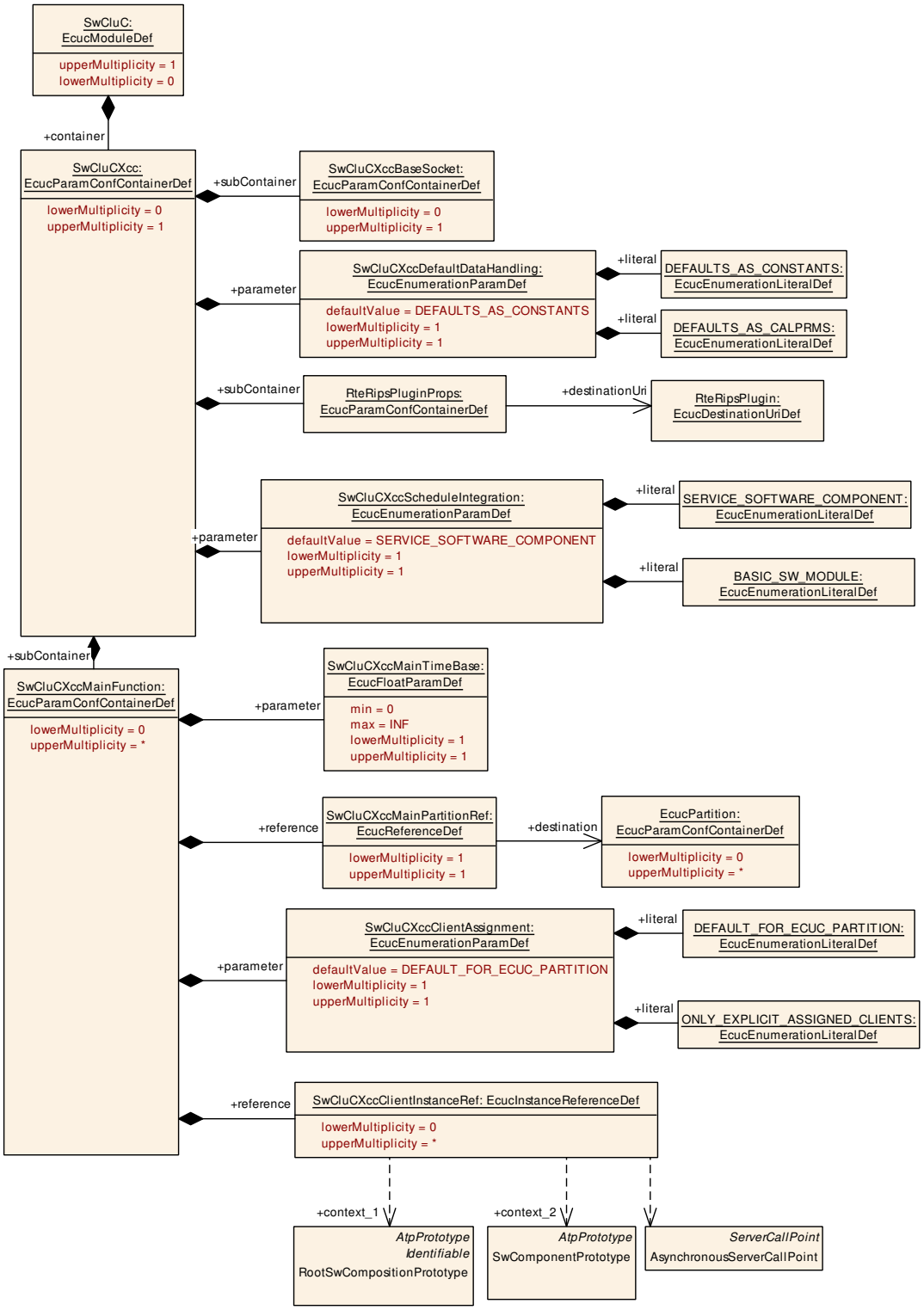

Figure 10.9: General Cross Cluster Communication Parameters

[ECUC_SwCluC_00075] Definition of EcucParamConfContainerDef SwCluCXcc ⌈

Container Name	SwCluCXcc
Parent Container	SwCluC
Upstream mapping defined	–
Description	Configuration of the Binary Manifest of the Software Cluster Connection
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCXccDefaultDataHandling	1	[ECUC_SwCluC_00084]
SwCluCXccScheduleIntegration	1	[ECUC_SwCluC_00156]

Included Containers		
Container Name	**Multiplicity**	**Description**
RteRipsPluginProps	1	This container defines the identity of the Rte Implementation Plug-in and provides the RTE relevant parameters of the Rte Implementation Plug-in. The shortName of the container defines the name of the Rte Implementation Plug-in used for the API infixes.
SwCluCXccBaseSocket	0..1	Maps a XccBaseSocket to the CpSoftwareClusterService Resource describing the XccBaseSocket resource in the clustered system. Maps a XccBaseSocket to a specific Ecuc Partition on which the XccBaseSocket is working.
SwCluCXccMainFunction	0..*	Defines the existence and properties of a SwCluC_Xcc_Main Function.

⌋

[ECUC_SwCluC_00084] Definition of EcucEnumerationParamDef SwCluCXccDefaultDataHandling ⌈

Parameter Name	SwCluCXccDefaultDataHandling	
Parent Container	SwCluCXcc	
Upstream mapping defined	–	
Description	SwCluCXccDefaultDataHandling defines whether the default data for unconnected RPortPrototypes of a Software Cluster are instantiated as fixed constants or as calibration parameter.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DEFAULTS_AS_CALPRMS	Default data are implemented as calibration parameter.
	DEFAULTS_AS_CONSTANTS	Default data are implemented as fixed constants.
Default value	DEFAULTS_AS_CONSTANTS	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00156] Definition of EcucEnumerationParamDef SwCluCXcc ScheduleIntegration ⌈

Parameter Name	SwCluCXccScheduleIntegration	
Parent Container	SwCluCXcc	
Upstream mapping defined	–	
Description	Defines whether the SwCluC_Xcc_MainFunction is integrated via a Service Software Component with RunnableEntity or a Basic Software Module Description with SchedulableEntity.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BASIC_SW_MODULE	SwCluC_Xcc_MainFunctions are integrated via a Basic Software Module Description with SchedulableEntity(s)
	SERVICE_SOFTWARE_COMPONENT	SwCluC_Xcc_MainFunctions are integrated via a Service Software Component with Runnable Entity(s)
Default value	SERVICE_SOFTWARE_COMPONENT	
Pre-Compile only	true	

⌋

10.2.6.1 Cross Cluster Communication Base Socket

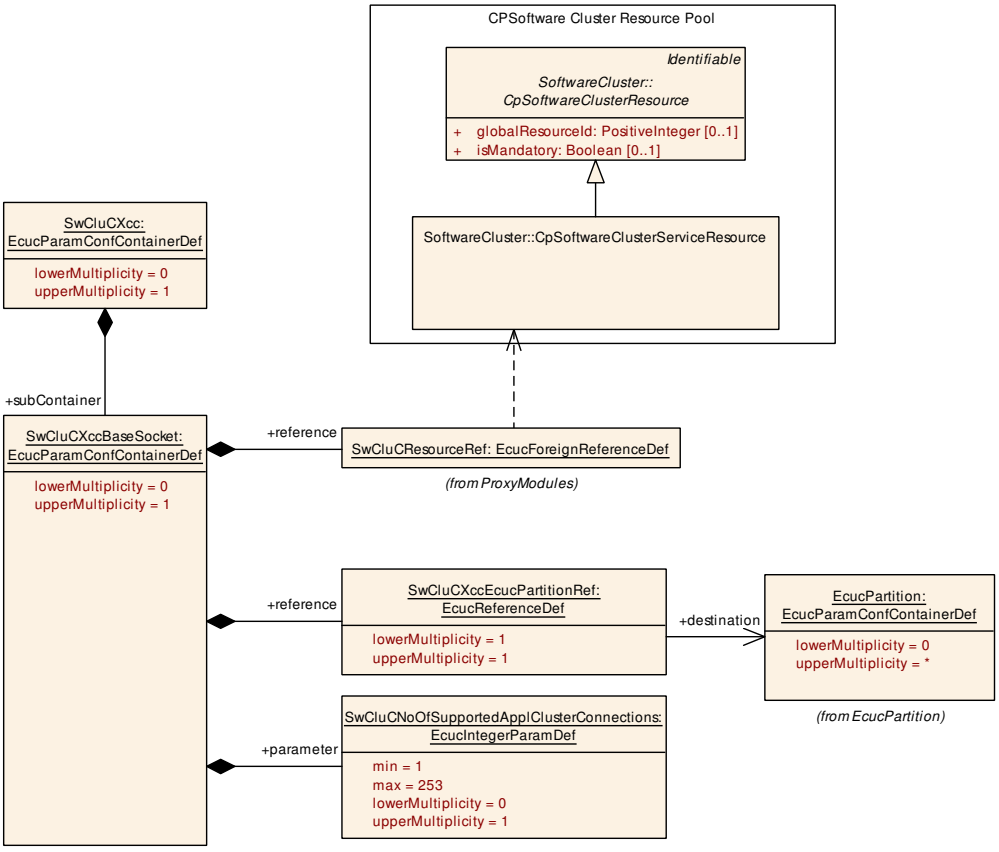

Figure 10.10: Cross Cluster Communication Base Socket

[ECUC_SwCluC_00076] Definition of EcucParamConfContainerDef SwCluCXccBaseSocket ⌈

Container Name	SwCluCXccBaseSocket
Parent Container	SwCluCXcc
Upstream mapping defined	–
Description	Maps a XccBaseSocket to the CpSoftwareClusterServiceResource describing the Xcc BaseSocket resource in the clustered system. Maps a XccBaseSocket to a specific EcucPartition on which the XccBaseSocket is working.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]
SwCluCXccEcucPartitionRef	1	[ECUC_SwCluC_00077]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00166] SwCluCNoOfSupportedApplClusterConnections, see definition below container SwCluCComProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[ECUC_SwCluC_00077] Definition of EcucReferenceDef SwCluCXccEcucPartitionRef ⌈

Parameter Name	SwCluCXccEcucPartitionRef
Parent Container	SwCluCXccBaseSocket
Upstream mapping defined	–
Description	Reference to the EcucPartition on which the XccBaseSocket is available.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_SwCluC_CONSTR_03063] `SwCluCXccBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_XCC_BASE_SOCKET`

Upstream requirements: SRS_SwCluC_00212, SRS_BSW_00167

⌈The `SwCluCXccBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_XCC_BASE_SOCKET`.⌋

10.2.6.2 Cross Cluster Communication Rte Plug-in Props

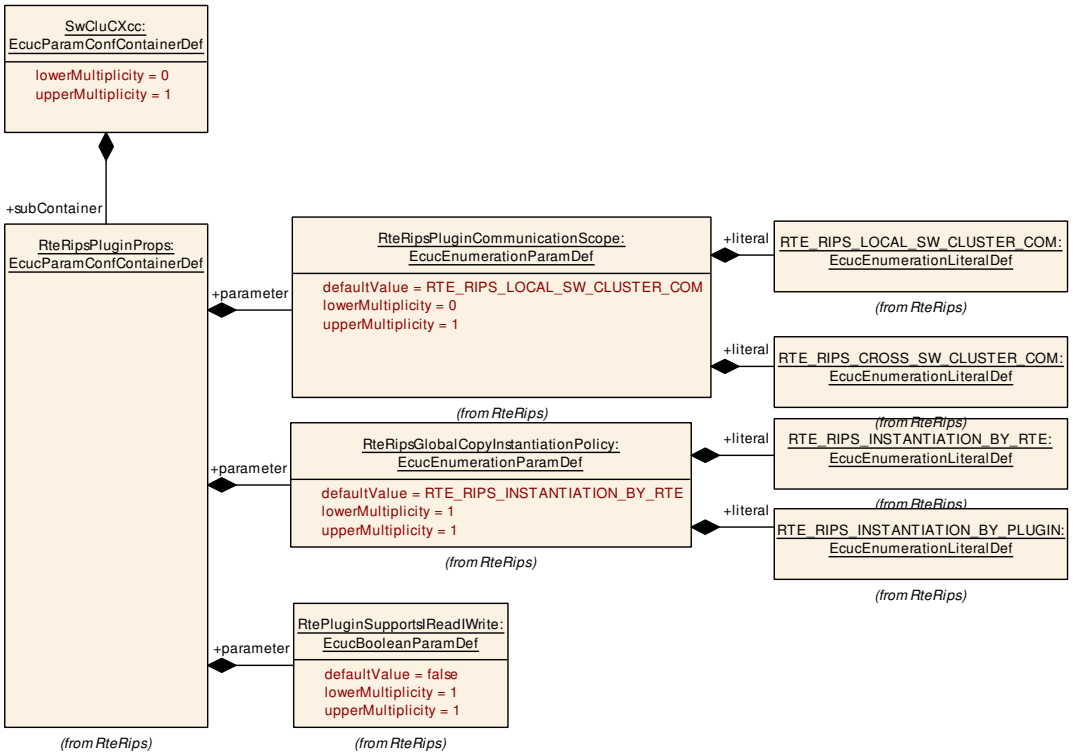

Figure 10.11: Cross Cluster Communication RTE Plug-In Properties

[ECUC_Rte_09173] Definition of EcucParamConfContainerDef RteRipsPlugin Props

Container Name	RteRipsPluginProps
Parent Container	SwCluCXcc
Parent Destination Uri Definition	RteRipsPlugin
Upstream mapping defined	–
Description	This container defines the identity of the Rte Implementation Plug-in and provides the RTE relevant parameters of the Rte Implementation Plug-in. The shortName of the container defines the name of the Rte Implementation Plug-in used for the API infixes.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RtePluginSupportsIReadIWrite	1	[ECUC_Rte_09169]
RteRipsGlobalCopyInstantiationPolicy	1	[ECUC_Rte_09170]
RteRipsPluginCommunicationScope	0..1	[ECUC_Rte_09171]

Included Containers		
Container Name	**Multiplicity**	**Description**
RteRipsPluginCommunicationTypeSupport	0..1	This container defines the different elements of communication graphs that the plugin is responsible for.

[ECUC_Rte_09169] Definition of EcucBooleanParamDef RtePluginSupportsIReadIWrite

Parameter Name	RtePluginSupportsIReadIWrite
Parent Container	RteRipsPluginProps
Upstream mapping defined	–
Description	Denotes if or if not the plug-in supports the Rte_Rips_IRead/IWrite macros for primitive data.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Rte_09170] Definition of EcucEnumerationParamDef RteRipsGlobalCopyInstantiationPolicy

Parameter Name	RteRipsGlobalCopyInstantiationPolicy	
Parent Container	RteRipsPluginProps	
Upstream mapping defined	–	
Description	Select whether for the relevant communication graph a RIPS plugin should instantiate the associated global copies itself or use global copies which have been instantiated by the RTE.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	RTE_RIPS_INSTANTIATION_BY_PLUGIN	For the relevant communication graphs, the RIPS plugin must provide the global copies of the associated data.
	RTE_RIPS_INSTANTIATION_BY_RTE	For the relevant communication graphs, the RTE must provide the global copies of the associated data.
Default value	RTE_RIPS_INSTANTIATION_BY_RTE	
Pre-Compile only	true	

[ECUC_Rte_09171] Definition of EcucEnumerationParamDef RteRipsPluginCommunicationScope

Parameter Name	RteRipsPluginCommunicationScope	
Parent Container	RteRipsPluginProps	
Upstream mapping defined	–	
Description	Defines the communication scope for which the Rte Implementation Plug-Ins (RIPS) serves. If this parameter is not set, the default behavior RTE_RIPS_LOCAL_SW_CLUSTER_COM applies.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	RTE_RIPS_CROSS_SW_CLUSTER_COM	The Rte Implementation Plug-In handles the Cross Software Cluster Communication.
	RTE_RIPS_LOCAL_SW_CLUSTER_COM	The Rte Implementation Plug-In handles the Local Software Cluster Communication.
Default value	RTE_RIPS_LOCAL_SW_CLUSTER_COM	
Pre-Compile only	true	

10.2.6.3 Cross Cluster Communication Main Function

[ECUC_SwCluC_00157] Definition of EcucParamConfContainerDef SwCluCXcc MainFunction ⌈

Container Name	SwCluCXccMainFunction
Parent Container	SwCluCXcc
Upstream mapping defined	–
Description	Defines the existence and properties of a SwCluC_Xcc_MainFunction.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCXccClientAssignment	1	[ECUC_SwCluC_00160]
SwCluCXccMainTimeBase	1	[ECUC_SwCluC_00158]
SwCluCXccClientInstanceRef	0..*	[ECUC_SwCluC_00161]
SwCluCXccMainPartitionRef	1	[ECUC_SwCluC_00159]

No Included Containers

⌋

[ECUC_SwCluC_00160] Definition of EcucEnumerationParamDef SwCluCXcc ClientAssignment ⌈

Parameter Name	SwCluCXccClientAssignment	
Parent Container	SwCluCXccMainFunction	
Upstream mapping defined	–	
Description	Defines if the SwCluC_Xcc_MainFunction is used as default main function for all clients which are not referenced explicitly by a SwCluCXccClientInstanceRef on this Ecuc Partition	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DEFAULT_FOR_ECUC_PARTITION	The related SwCluC_Xcc_MainFunction is used as default main function to handle timeout for all clients which are not referenced explicitly by a SwCluCXccClientInstanceRef on this EcucPartition.
	ONLY_EXPLICIT_ASSIGNED_CLIENTS	The related SwCluC_Xcc_MainFunction only handles timeout for clients which are explicitly referenced by a SwCluCXccClientInstanceRef by this SwCluCXccMainFunction.
Default value	DEFAULT_FOR_ECUC_PARTITION	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00158] Definition of EcucFloatParamDef SwCluCXccMainTimeBase ⌈

Parameter Name	SwCluCXccMainTimeBase
Parent Container	SwCluCXccMainFunction
Upstream mapping defined	–
Description	The period between successive calls to according instance of SwCluC_Xcc_MainFunction in seconds. This parameter may be used by the SwCluC generator to transform the given timeout values to internal implementation specific counter or tick values.
Multiplicity	1
Type	EcucFloatParamDef
Range	[0 .. INF]
Default value	–
Pre-Compile only	true

⌋

[ECUC_SwCluC_00161] Definition of EcucInstanceReferenceDef SwCluCXccClientInstanceRef ⌈

Parameter Name	SwCluCXccClientInstanceRef
Parent Container	SwCluCXccMainFunction
Upstream mapping defined	–
Description	Reference to an AsynchronousServerCallPoint
Multiplicity	0..*
Type	Instance reference to ASYNCHRONOUS-SERVER-CALL-POINT context: ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE
Pre-Compile only	true

⌋

[ECUC_SwCluC_00159] Definition of EcucReferenceDef SwCluCXccMainPartitionRef ⌈

Parameter Name	SwCluCXccMainPartitionRef
Parent Container	SwCluCXccMainFunction
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according SwCluC_Xcc_MainFunction instance is assigned to.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_SwCluC_CONSTR_03403]

Upstream requirements: SRS_SwCluC_00100, SRS_SwCluC_00103, SRS_BSW_00167

⌈It is only allowed to configure at most one `SwCluCXccMainFunction` where `SwCluCXccClientAssignment` is set to `DEFAULT_FOR_ECUC_PARTITION` per `EcucPartition`.⌋

10.2.7 Proxy Modules

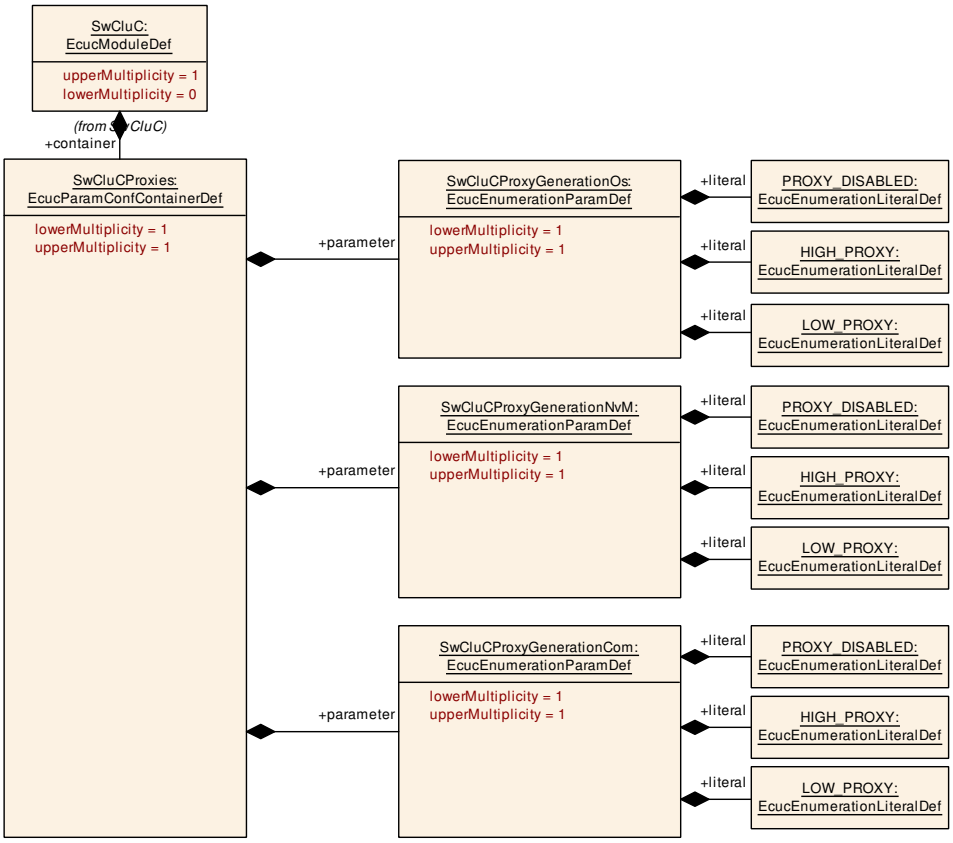

Figure 10.12: General Proxies Parameters

[ECUC_SwCluC_00036] Definition of EcucParamConfContainerDef SwCluCProxies ⌈

Container Name	SwCluCProxies
Parent Container	SwCluC
Upstream mapping defined	–
Description	General configuration of the Proxy Modules of Software Cluster Connection
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCProxyGenerationCom	1	[ECUC_SwCluC_00100]
SwCluCProxyGenerationDcm	1	[ECUC_SwCluC_00102]
SwCluCProxyGenerationDem	1	[ECUC_SwCluC_00103]
SwCluCProxyGenerationFiM	1	[ECUC_SwCluC_00104]
SwCluCProxyGenerationNvM	1	[ECUC_SwCluC_00038]
SwCluCProxyGenerationOs	1	[ECUC_SwCluC_00037]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCComProxy	0..1	Com Proxy specific parameters.
SwCluCDcmProxy	0..1	Dcm Proxy specific parameters.
SwCluCDemProxy	0..1	DemProxy specific parameters.
SwCluCFiMProxy	0..1	FiM Proxy specific parameters.
SwCluCNvMProxy	0..1	NvM Proxy specific parameters.
SwCluCOsProxy	0..1	Os Proxy specific parameters.

⌋

[ECUC_SwCluC_00100] Definition of EcucEnumerationParamDef SwCluCProxyGenerationCom ⌈

Parameter Name	SwCluCProxyGenerationCom	
Parent Container	SwCluCProxies	
Upstream mapping defined	–	
Description	Defines whether the Com Proxy code and models are generated.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.
	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation.
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00102] Definition of EcucEnumerationParamDef SwCluCProxyGenerationDcm ⌈

Parameter Name	SwCluCProxyGenerationDcm	
Parent Container	SwCluCProxies	
Upstream mapping defined	–	
Description	Defines whether the Dcm Proxy code and models are generated.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.
	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation.
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00103] Definition of EcucEnumerationParamDef SwCluCProxyGenerationDem ⌈

Parameter Name	SwCluCProxyGenerationDem	
Parent Container	SwCluCProxies	
Upstream mapping defined	–	
Description	Defines whether the Dem Proxy code and models are generated.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.
	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation.
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00104] Definition of EcucEnumerationParamDef SwCluCProxyGenerationFiM ⌈

Parameter Name	SwCluCProxyGenerationFiM	
Parent Container	SwCluCProxies	
Upstream mapping defined	–	
Description	Defines whether the FiM Proxy code and models are generated.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.
	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation.
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00038] Definition of EcucEnumerationParamDef SwCluCProxyGenerationNvM ⌈

Parameter Name	SwCluCProxyGenerationNvM	
Parent Container	SwCluCProxies	
Upstream mapping defined	–	
Description	Defines whether the NvM Proxy code and models are generated.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.

▽

△

	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation.
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00037] Definition of EcucEnumerationParamDef SwCluCProxyGenerationOs ⌈

Parameter Name	SwCluCProxyGenerationOs	
Parent Container	SwCluCProxies	
Upstream mapping defined	–	
Description	Defines whether the Os Proxy code and models are generated.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.
	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation
Pre-Compile only	true	

⌋

10.2.7.1 NvM Proxy

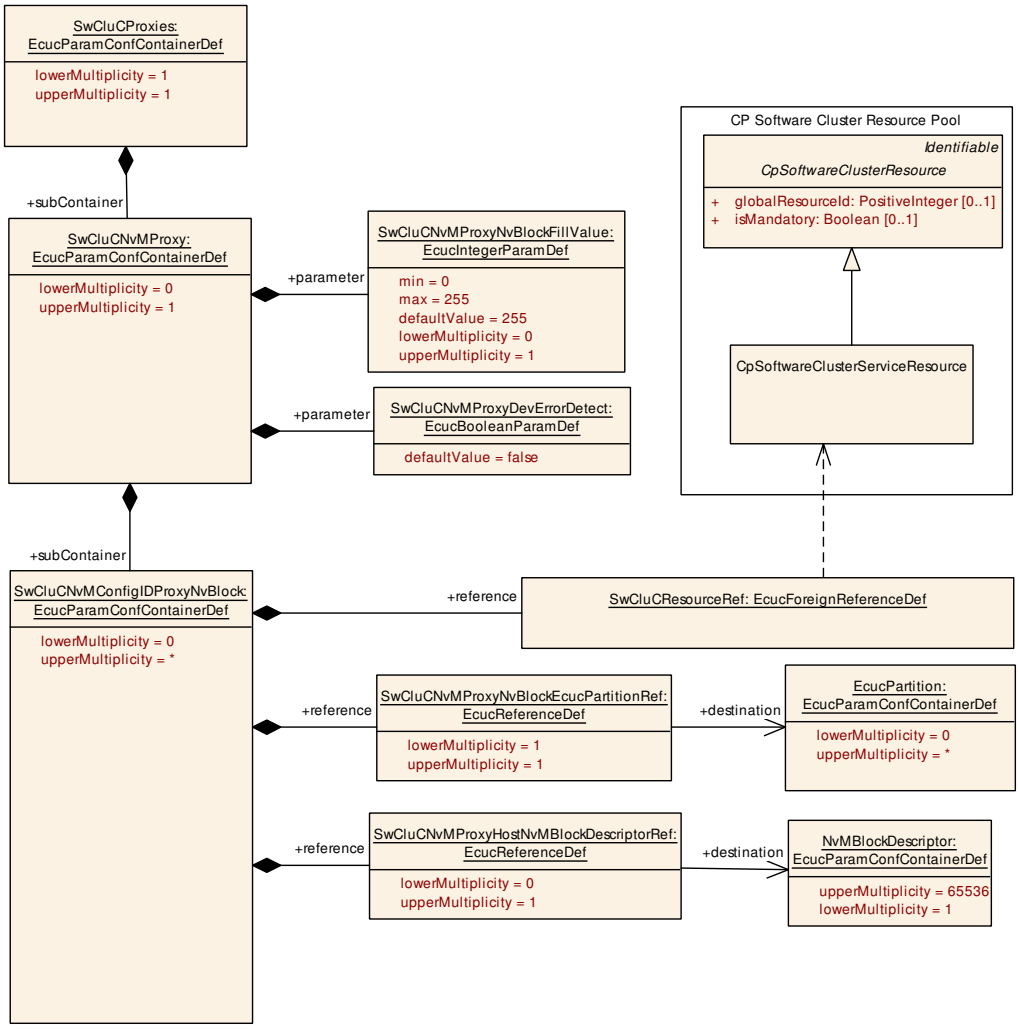

Figure 10.13: NvM Proxy Parameter

[ECUC_SwCluC_00039] Definition of EcucParamConfContainerDef SwCluCNvMProxy ⌈

Container Name	SwCluCNvMProxy
Parent Container	SwCluCProxies
Upstream mapping defined	–
Description	NvM Proxy specific parameters.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNvMProxyDevErrorDetect	1	[ECUC_SwCluC_00083]
SwCluCNvMProxyNvBlockFillValue	0..1	[ECUC_SwCluC_00055]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCNvMBaseSocket	1..*	This container configures how many EcucPartitions specific API links are required for the NvM Proxy and to which cluster resource the API set belongs. In the NvM Low Proxy all offered API sets needs to be configured. In the NvM High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the Ecuc Partition on which the Nv Block is handled.
SwCluCNvMConfigIDProxyNvBlock	0..*	Describes a Configuration ID block of an Application Software Cluster and maps it to • a CpSoftwareClusterServiceResource describing the NvBlock resource in the clustered system • a EcucPartition • a NvMBlockDescriptor (only in the context of the Host Software Cluster) In the scope of an Application Software Cluster at most one SwCluCNvMConfigIDProxyNvBlock container can exist. In the scope of a Host Software Cluster multiple SwCluCNvMConfigIDProxyNvBlock container can exist, one for each Configuration ID block of Application Software Clusters.
SwCluCNvMProxyNvBlock	1..*	Maps a NvMBlockDescriptor to the CpSoftwareClusterService Resource describing the NvBlock resource in the clustered system.

⌋

[ECUC_SwCluC_00083] Definition of EcucBooleanParamDef SwCluCNvMProxyDevErrorDetect ⌈

Parameter Name	SwCluCNvMProxyDevErrorDetect
Parent Container	SwCluCNvMProxy
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_SwCluC_00055] Definition of EcucIntegerParamDef SwCluCNvMProxyNvBlockFillValue ⌈

Parameter Name	SwCluCNvMProxyNvBlockFillValue
Parent Container	SwCluCNvMProxy
Upstream mapping defined	–

▽

△

Description	Defines the default value used to fill unused bytes in NvBlock space in case NvBlock length in Application Software Cluster < NvBlock length configured in NvM of Host Software Cluster. Dependent on NvM Proxy implementation this parameter is only relevant for NvM Low Proxy or NvM High Proxy	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	255	
Pre-Compile only	true	

⌋

10.2.7.1.1 NvM Base Socket

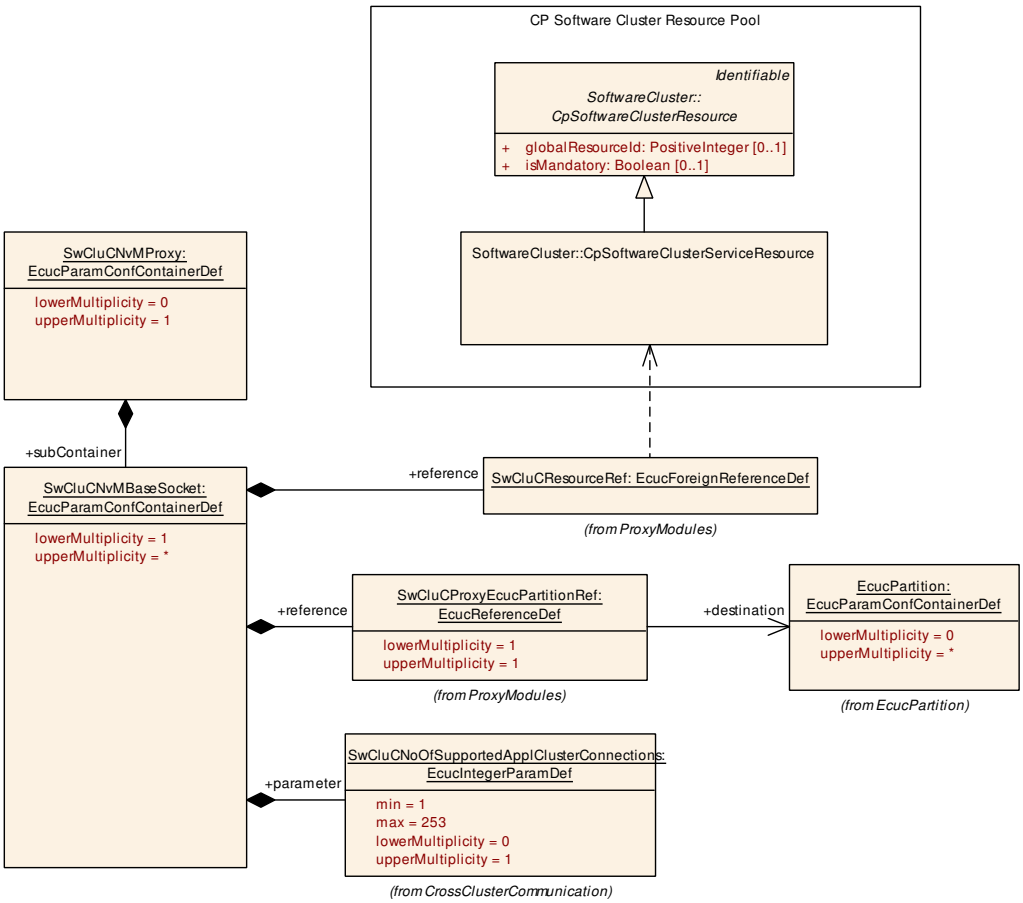

Figure 10.14: NvM Proxy Base Socket

[ECUC_SwCluC_00056] Definition of EcucParamConfContainerDef SwCluCNvMBaseSocket ⌈

Container Name	SwCluCNvMBaseSocket
Parent Container	SwCluCNvMProxy
Upstream mapping defined	–
Description	This container configures how many EcucPartitions specific API links are required for the NvM Proxy and to which cluster resource the API set belongs. In the NvM Low Proxy all offered API sets needs to be configured. In the NvM High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the Nv Block is handled.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00166] SwCluCNoOfSupportedApplClusterConnections, see definition below container SwCluCComProxyBaseSocket.

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02130] **`SwCluCNvMBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_NVM_BASE_SOCKET`**

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00167

⌈The `SwCluCNvMBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_NVM_BASE_SOCKET`.⌋

10.2.7.1.2 NvM Proxy NvBlock configuration

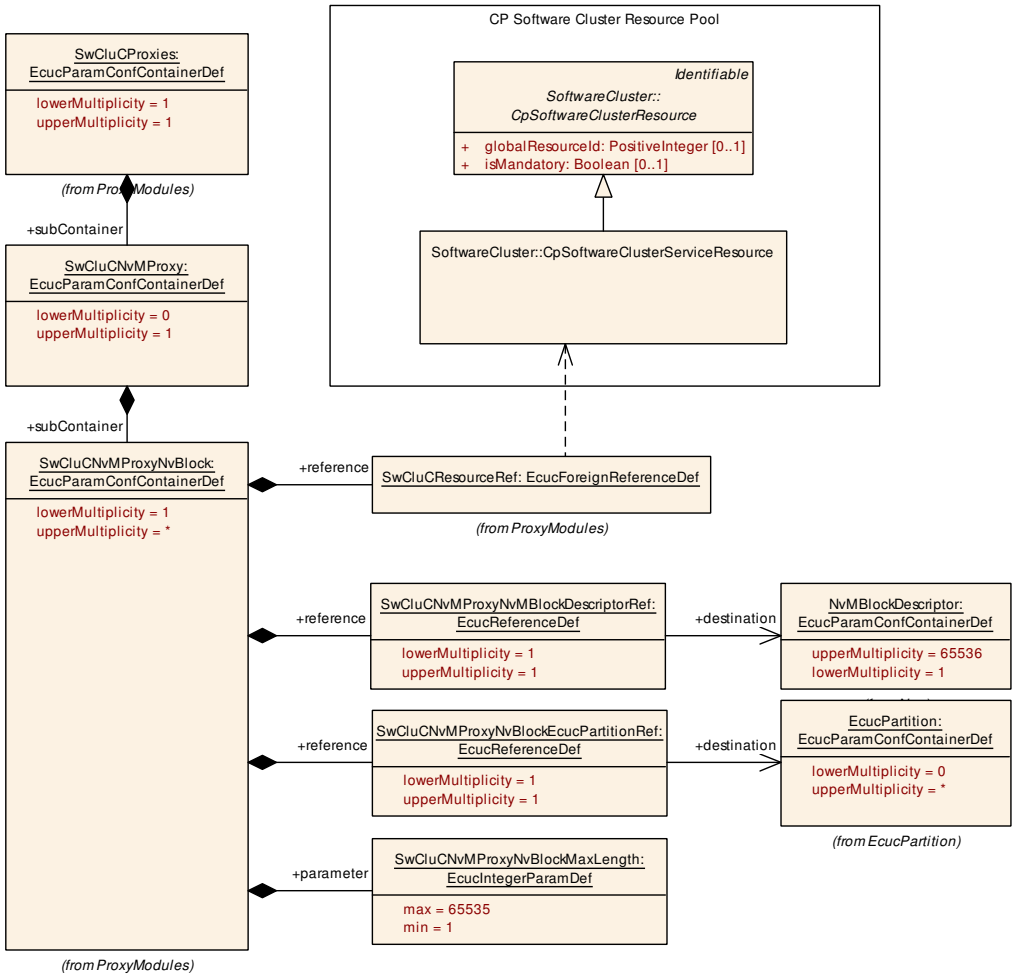

Figure 10.15: NvM Proxy NvBlock

[ECUC_SwCluC_00040] Definition of EcucParamConfContainerDef SwCluCNvMProxyNvBlock ⌈

Container Name	SwCluCNvMProxyNvBlock
Parent Container	SwCluCNvMProxy
Upstream mapping defined	–
Description	Maps a NvMBlockDescriptor to the CpSoftwareClusterServiceResource describing the NvBlock resource in the clustered system.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNvMProxyNvBlockMaxLength	1	[ECUC_SwCluC_00043]
SwCluCNvMProxyNvBlockEcucPartitionRef	1	[ECUC_SwCluC_00060]
SwCluCNvMProxyNvMBlockDescriptorRef	1	[ECUC_SwCluC_00042]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00043] Definition of EcucIntegerParamDef SwCluCNvMProxyNvBlockMaxLength ⌈

Parameter Name	SwCluCNvMProxyNvBlockMaxLength	
Parent Container	SwCluCNvMProxyNvBlock	
Upstream mapping defined	–	
Description	Defines the maximum NV block data length in bytes.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

For parameter table [ECUC_SwCluC_00060] SwCluCNvMProxyNvBlockEcucPartitionRef, see definition below container SwCluCNvMConfigIDProxyNvBlock.

[ECUC_SwCluC_00042] Definition of EcucReferenceDef SwCluCNvMProxyNvMBlockDescriptorRef ⌈

Parameter Name	SwCluCNvMProxyNvMBlockDescriptorRef
Parent Container	SwCluCNvMProxyNvBlock
Upstream mapping defined	–
Description	Reference to the NvMBlockDescriptor
Multiplicity	1
Type	Reference to NvMBlockDescriptor
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02131] `SwCluCNvMProxyNvBlock` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_NV_BLOCK`

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00167

⌈The `SwCluCNvMProxyNvBlock.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_NV_BLOCK`.⌋

10.2.7.1.2.1 NvM Proxy Configuration ID Block configuration

[ECUC_SwCluC_00162] Definition of EcucParamConfContainerDef SwCluCNvMConfigIDProxyNvBlock ⌈

Container Name	SwCluCNvMConfigIDProxyNvBlock
Parent Container	SwCluCNvMProxy
Upstream mapping defined	–
Description	Describes a Configuration ID block of an Application Software Cluster and maps it to • a CpSoftwareClusterServiceResource describing the NvBlock resource in the clustered system • a EcucPartition • a NvMBlockDescriptor (only in the context of the Host Software Cluster) In the scope of an Application Software Cluster at most one SwCluCNvMConfig IDProxyNvBlock container can exist. In the scope of a Host Software Cluster multiple SwCluCNvMConfigIDProxyNvBlock container can exist, one for each Configuration ID block of Application Software Clusters.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNvMProxyHostNvMBlockDescriptorRef	0..1	[ECUC_SwCluC_00163]
SwCluCNvMProxyNvBlockEcucPartitionRef	1	[ECUC_SwCluC_00060]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00163] Definition of EcucReferenceDef SwCluCNvMProxyHostNvMBlockDescriptorRef ⌈

Parameter Name	SwCluCNvMProxyHostNvMBlockDescriptorRef
Parent Container	SwCluCNvMConfigIDProxyNvBlock
Upstream mapping defined	–
Description	Reference to the NvMBlockDescriptor of the NvM module in Host Software Cluster. This reference is only applicable in the context of a Host Software Cluster.
Multiplicity	0..1
Type	Reference to NvMBlockDescriptor
Pre-Compile only	true

⌋

[ECUC_SwCluC_00060] Definition of EcucReferenceDef SwCluCNvMProxyNvBlockEcucPartitionRef ⌈

Parameter Name	SwCluCNvMProxyNvBlockEcucPartitionRef
Parent Container	SwCluCNvMConfigIDProxyNvBlock, SwCluCNvMProxyNvBlock
Upstream mapping defined	–
Description	Reference to the EcucPartition on which the NvM Proxy offers the NvBlock access (call of APIfunctions, callback functions and Ports on the Service Software Component)
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02158]

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00167

⌈In case the parameter `NvMDynamicConfiguration` is set to `true` in the NvM module configuration of the `Application Software Cluster`, exactly one container `SwCluCNvMConfigIDProxyNvBlock` shall exist in the scope of the same `Application Software Cluster`.⌋

[SWS_SwCluC_CONSTR_02159]

Upstream requirements: SRS_SwCluC_00206, SRS_BSW_00167

⌈In the scope of an `Application Software Cluster` at most one `SwCluCNvMConfigIDProxyNvBlock` container shall exist.⌋

10.2.7.2 Os Proxy

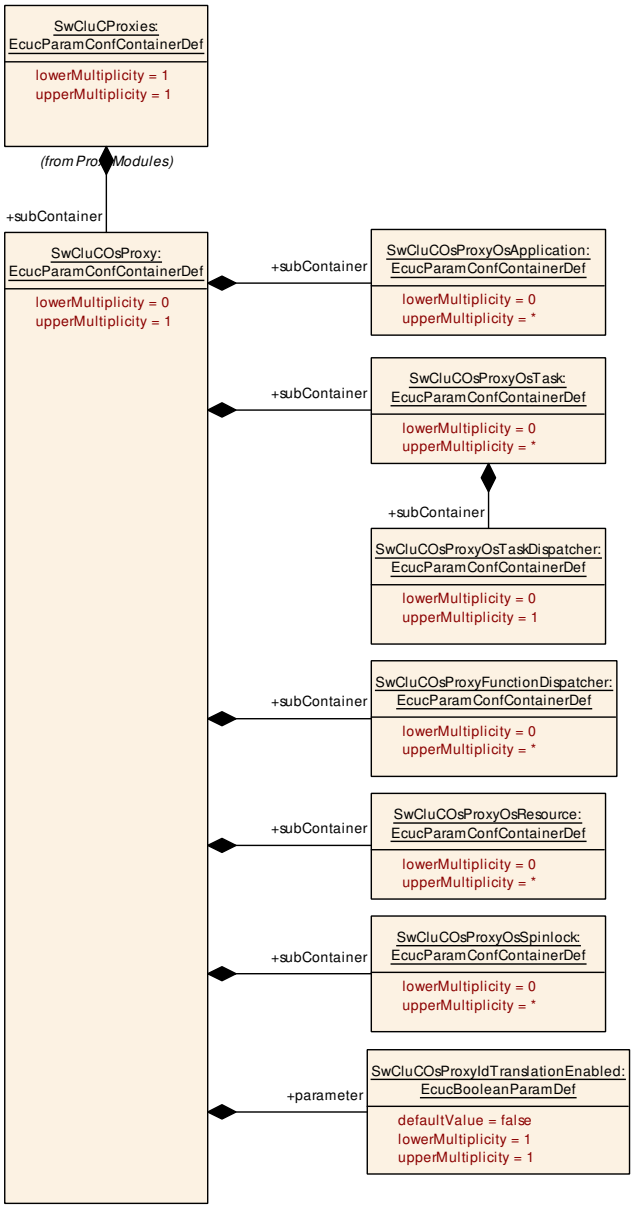

Figure 10.16: Os Proxy Parameter

[ECUC_SwCluC_00085] Definition of EcucParamConfContainerDef SwCluCOs Proxy ⌈

Container Name	SwCluCOsProxy
Parent Container	SwCluCProxies
Upstream mapping defined	–
Description	Os Proxy specific parameters.
Multiplicity	0..1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyIdTranslationEnabled	1	[ECUC_SwCluC_00155]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCOsProxyFunction Dispatcher	0..*	This container configures a function dispatcher. In the Os High Proxy, this configuration creates a Dispatch Entry Point in the Binary Manifest, which can be called by the Os Low Proxy. In the Os Low Proxy, this configuration provides the Dispatcher Function (and according resource Entry in the Binary Manifest), which is able to call the MaxNumber OfCallee Dispatch Entry Points of Os High Proxies.
SwCluCOsProxyOsApplication	0..*	Maps a OsApplication to the CpSoftwareClusterService Resource describing the OsApplication resource in the clustered system.
SwCluCOsProxyOsBaseSocket	0..*	Maps a OsBaseSocket to the CpSoftwareClusterService Resource describing the OsBaseSocket resource in the clustered system.
SwCluCOsProxyOsResource	0..*	Maps a OsResource to the CpSoftwareClusterServiceResource describing the OsResource resource in the clustered system.
SwCluCOsProxyOsSpinlock	0..*	Maps a OsSpinlock to the CpSoftwareClusterServiceResource describing the OsSpinlock resource in the clustered system.
SwCluCOsProxyOsTask	0..*	Maps a OsTask to the CpSoftwareClusterServiceResource describing the OsTask resource in the clustered system.

⌋

[ECUC_SwCluC_00155] Definition of EcucBooleanParamDef SwCluCOsProxyId TranslationEnabled ⌈

Parameter Name	SwCluCOsProxyIdTranslationEnabled
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	Enables the translation of core IDs and application IDs between Host SW Cluster and Application SW Cluster - true: IDs translation from HOST into Application SW Cluster specific ones is enabled. - false: IDs translation from HOST into Application SW Cluster specific ones is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

10.2.7.2.1 Os Proxy Base Socket configuration

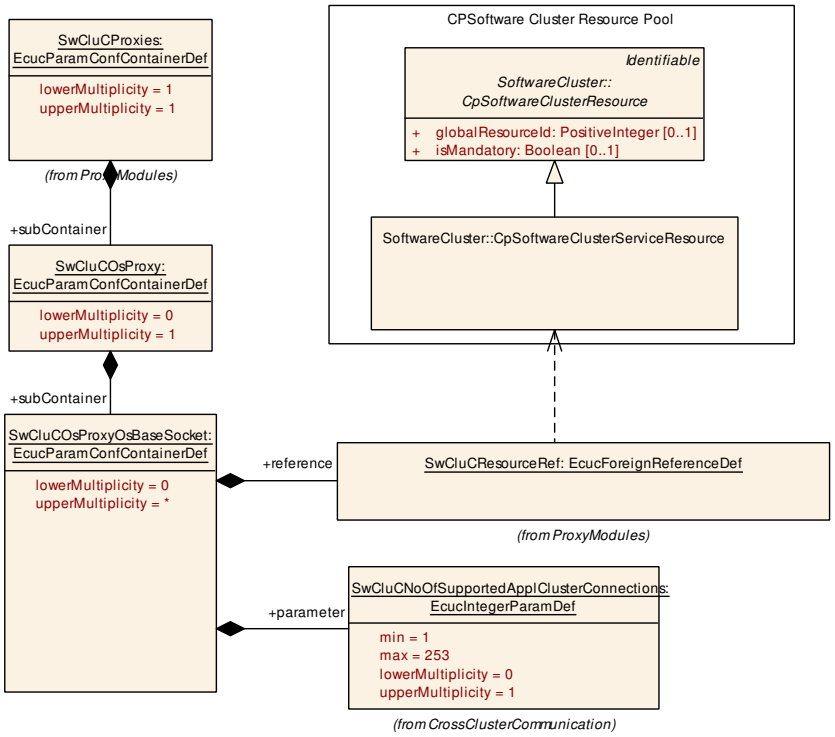

Figure 10.17: Os Proxy Base Socket Parameter

[ECUC_SwCluC_00053] Definition of EcucParamConfContainerDef SwCluCOsProxyOsBaseSocket ⌈

Container Name	SwCluCOsProxyOsBaseSocket
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	Maps a OsBaseSocket to the CpSoftwareClusterServiceResource describing the Os BaseSocket resource in the clustered system.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00166] SwCluCNoOfSupportedApplClusterConnections, see definition below container SwCluCComProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02231] `SwCluCOsProxyOsBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_BASE_SOCKET`

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00167

⌈The `SwCluCOsProxyOsBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_BASE_SOCKET`.⌋

10.2.7.2.2 Os Proxy OsApplication configuration

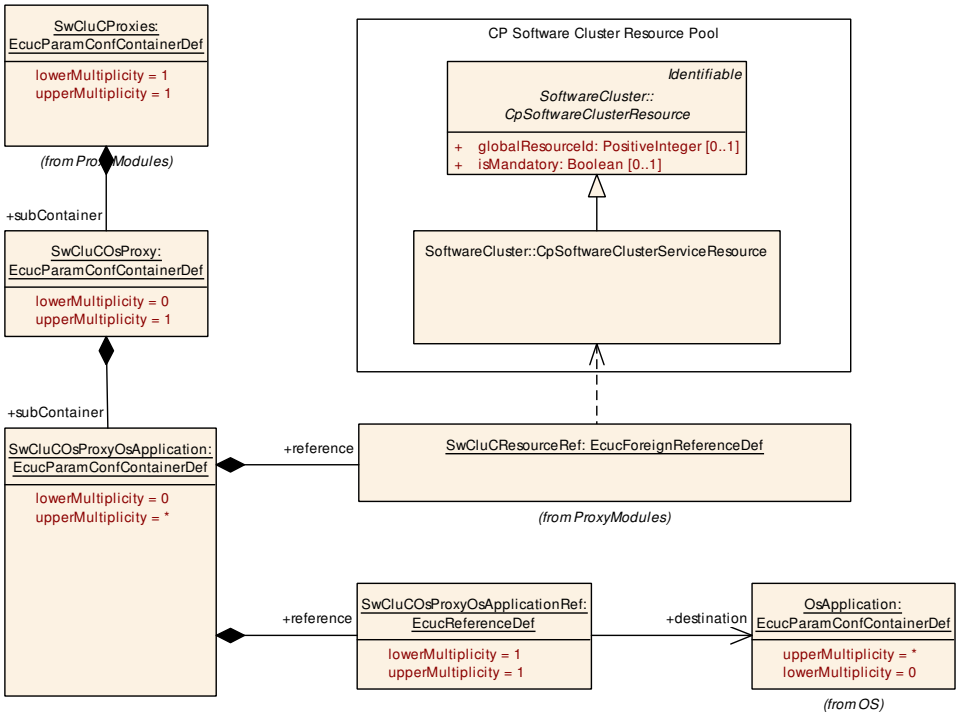

Figure 10.18: Os Proxy OsApplication Parameter

[ECUC_SwCluC_00046] Definition of EcucParamConfContainerDef SwCluCOsProxyOsApplication ⌈

Container Name	SwCluCOsProxyOsApplication
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	Maps a OsApplication to the CpSoftwareClusterServiceResource describing the Os Application resource in the clustered system.
Multiplicity	0..*

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyOsApplicationRef	1	[ECUC_SwCluC_00050]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00050] Definition of EcucReferenceDef SwCluCOsProxyOsApplicationRef ⌈

Parameter Name	SwCluCOsProxyOsApplicationRef
Parent Container	SwCluCOsProxyOsApplication
Upstream mapping defined	–
Description	Reference to the OsApplication
Multiplicity	1
Type	Reference to OsApplication
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02232] `SwCluCOsProxyOsApplication` relates only to a `CpSoftwareClusterServiceResource` of `category` `SWCLUSTER_RES_OS_APPLICATION`

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00167

⌈The `SwCluCOsProxyOsApplication.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_APPLICATION`.⌋

10.2.7.2.3 Os Proxy OsTask configuration

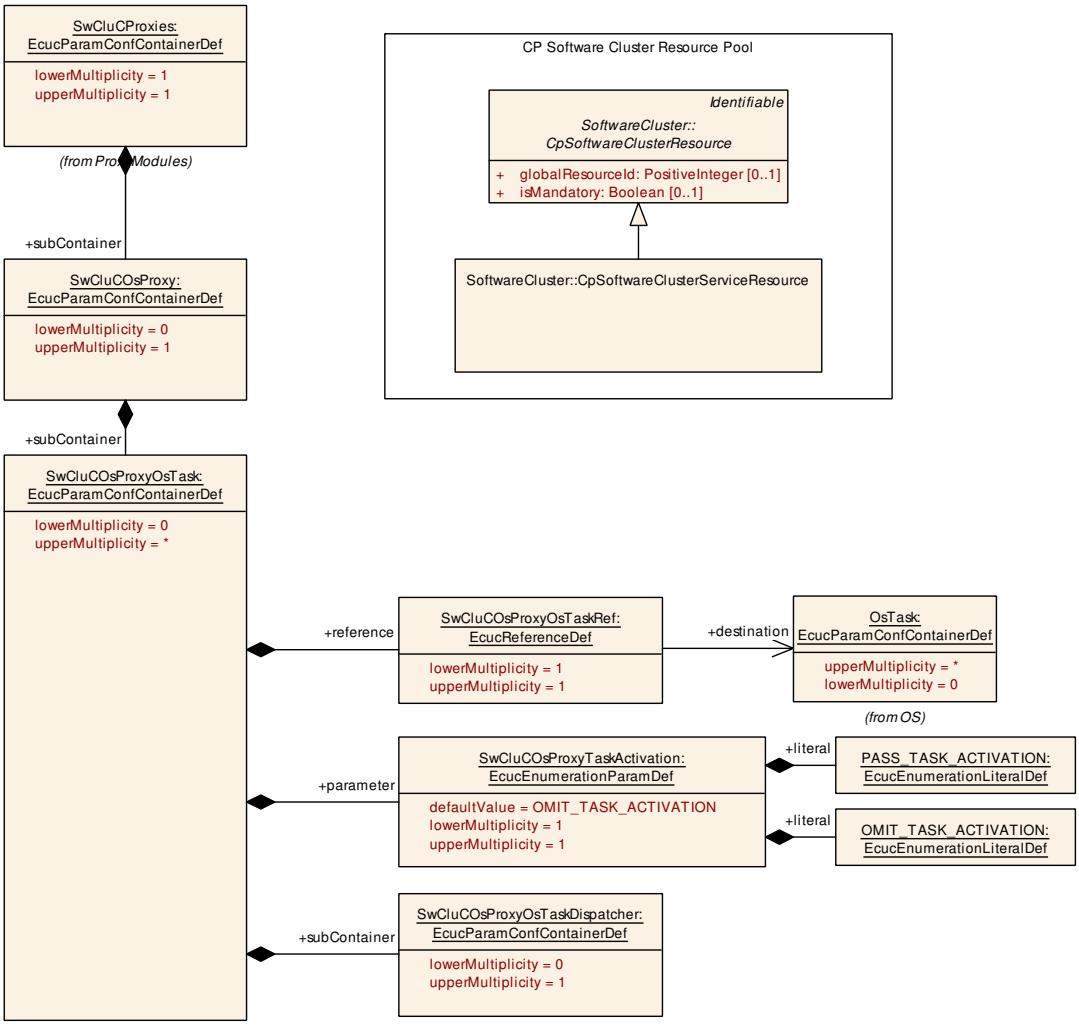

Figure 10.19: Os Proxy OsTask Parameter

[ECUC_SwCluC_00044] Definition of EcucParamConfContainerDef SwCluCOs ProxyOsTask ⌈

Container Name	SwCluCOsProxyOsTask
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	Maps a OsTask to the CpSoftwareClusterServiceResource describing the OsTask resource in the clustered system.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyTaskActivation	1	[ECUC_SwCluC_00048]
SwCluCOsProxyOsTaskRef	1	[ECUC_SwCluC_00045]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCOsProxyOsTaskDispatcher	0..1	This container defines a task dispatcher for the owning proxy task in the Os Low Proxy. In the Os High Proxy, this configuration is not used. In the Os Low Proxy, this configuration provides the Dispatcher Runnable. This will cause a call to the SwCluCOs ProxyOsTask(s) in the Os High Proxy. The Os Low Proxy is able to call one OsProxyOsTask in each of the MaxNumberOfCallee Os High Proxies. If none of the SwCluCOsProxy Dispatcher<xxx>Event containers is configured it's up to the integrator to model the appropriate RTEEvent.

⌋

[ECUC_SwCluC_00048] Definition of EcucEnumerationParamDef SwCluCOs ProxyTaskActivation ⌈

Parameter Name	SwCluCOsProxyTaskActivation	
Parent Container	SwCluCOsProxyOsTask	
Upstream mapping defined	–	
Description	Configures for this particular OsTask whether ActivateTask is passed to the Host Software Cluster or not.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	OMIT_TASK_ACTIVATION	ActivateTask is NOT passed to the Host Software Cluster.
	PASS_TASK_ACTIVATION	ActivateTask is passed to the Host Software Cluster.
Default value	OMIT_TASK_ACTIVATION	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00045] Definition of EcucReferenceDef SwCluCOsProxyOsTask Ref ⌈

Parameter Name	SwCluCOsProxyOsTaskRef
Parent Container	SwCluCOsProxyOsTask
Upstream mapping defined	–
Description	Reference to the OsTask
Multiplicity	1
Type	Reference to OsTask
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02233] **SwCluCOsProxyOsTask relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_TASK**

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00167

⌈The SwCluCOsProxyOsTask.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_TASK.⌋

10.2.7.2.4 Os Proxy OsResorce configuration

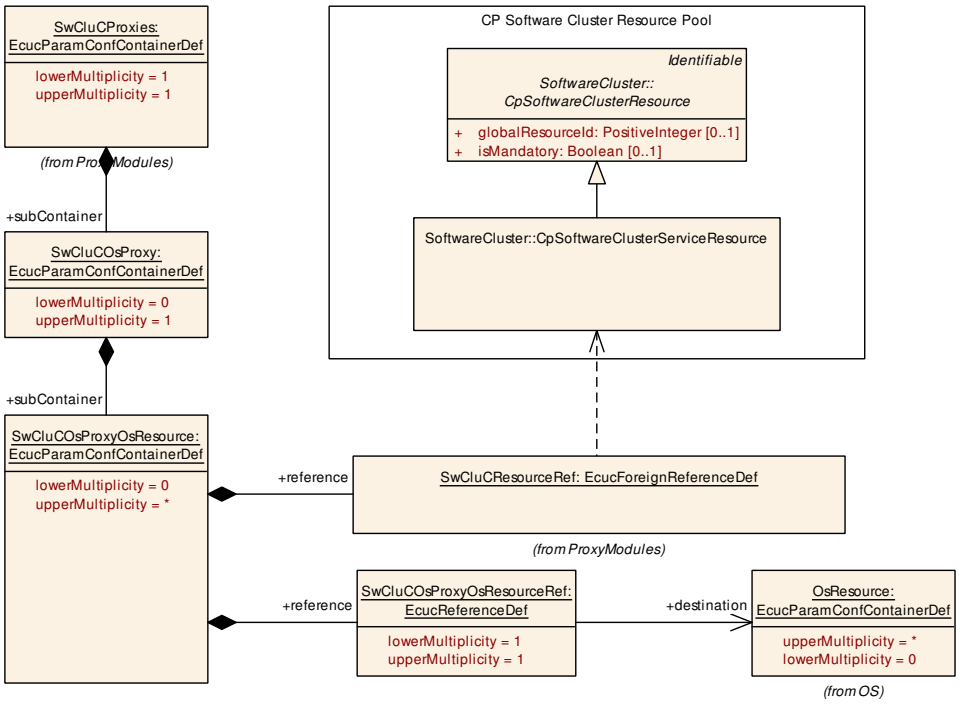

Figure 10.20: Os Proxy OsResource Parameter

[ECUC_SwCluC_00049] Definition of EcucParamConfContainerDef SwCluCOsProxyOsResource ⌈

Container Name	SwCluCOsProxyOsResource
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	Maps a OsResource to the CpSoftwareClusterServiceResource describing the Os Resource resource in the clustered system.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyOsResourceRef	1	[ECUC_SwCluC_00047]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00047] Definition of EcucReferenceDef SwCluCOsProxyOsResourceRef ⌈

Parameter Name	SwCluCOsProxyOsResourceRef
Parent Container	SwCluCOsProxyOsResource
Upstream mapping defined	–
Description	Reference to the OsResource
Multiplicity	1
Type	Reference to OsResource
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02234] `SwCluCOsProxyOsResource` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_RESOURCE`

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00167

⌈The `SwCluCOsProxyOsResource.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_RESOURCE`.⌋

10.2.7.2.5 Os Proxy OsSpinlock configuration

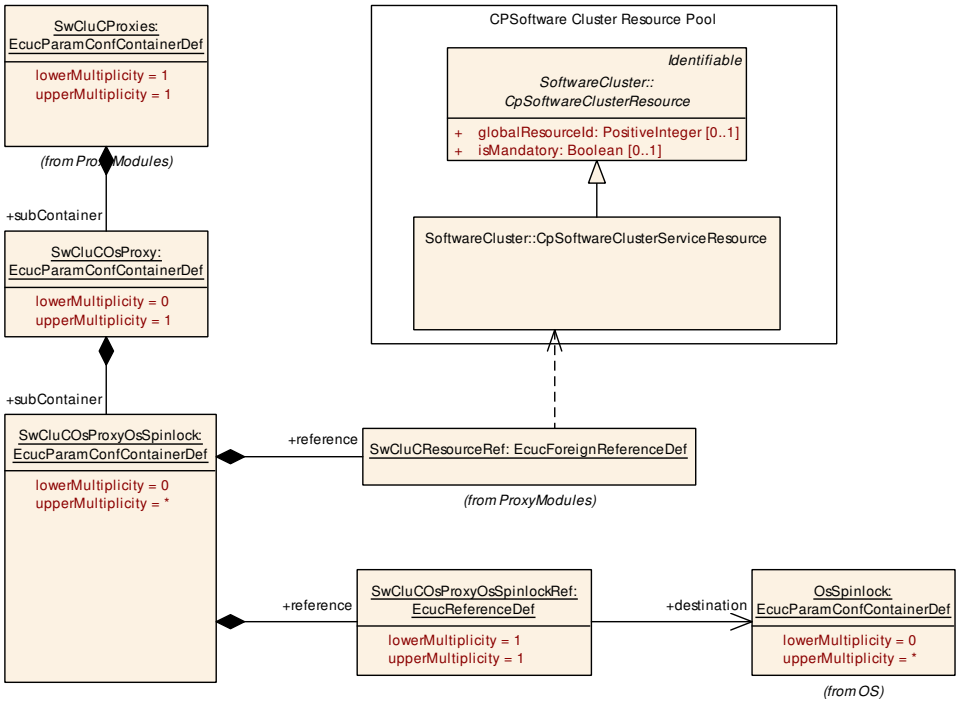

Figure 10.21: Os Proxy OsSpinlock Parameter

[ECUC_SwCluC_00051] Definition of EcucParamConfContainerDef SwCluCOs ProxyOsSpinlock ⌈

Container Name	SwCluCOsProxyOsSpinlock
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	Maps a OsSpinlock to the CpSoftwareClusterServiceResource describing the Os Spinlock resource in the clustered system.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyOsSpinlockRef	1	[ECUC_SwCluC_00052]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00052] Definition of EcucReferenceDef SwCluCOsProxyOsSpinlockRef ⌈

Parameter Name	SwCluCOsProxyOsSpinlockRef
Parent Container	SwCluCOsProxyOsSpinlock
Upstream mapping defined	–
Description	Reference to the OsSpinlock
Multiplicity	1
Type	Reference to OsSpinlock
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02235] `SwCluCOsProxyOsSpinlock` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_SPINLOCK`

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00167

⌈The `SwCluCOsProxyOsSpinlock.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_SPINLOCK`.⌋

10.2.7.2.6 Os Proxy Task Dispatcher configuration

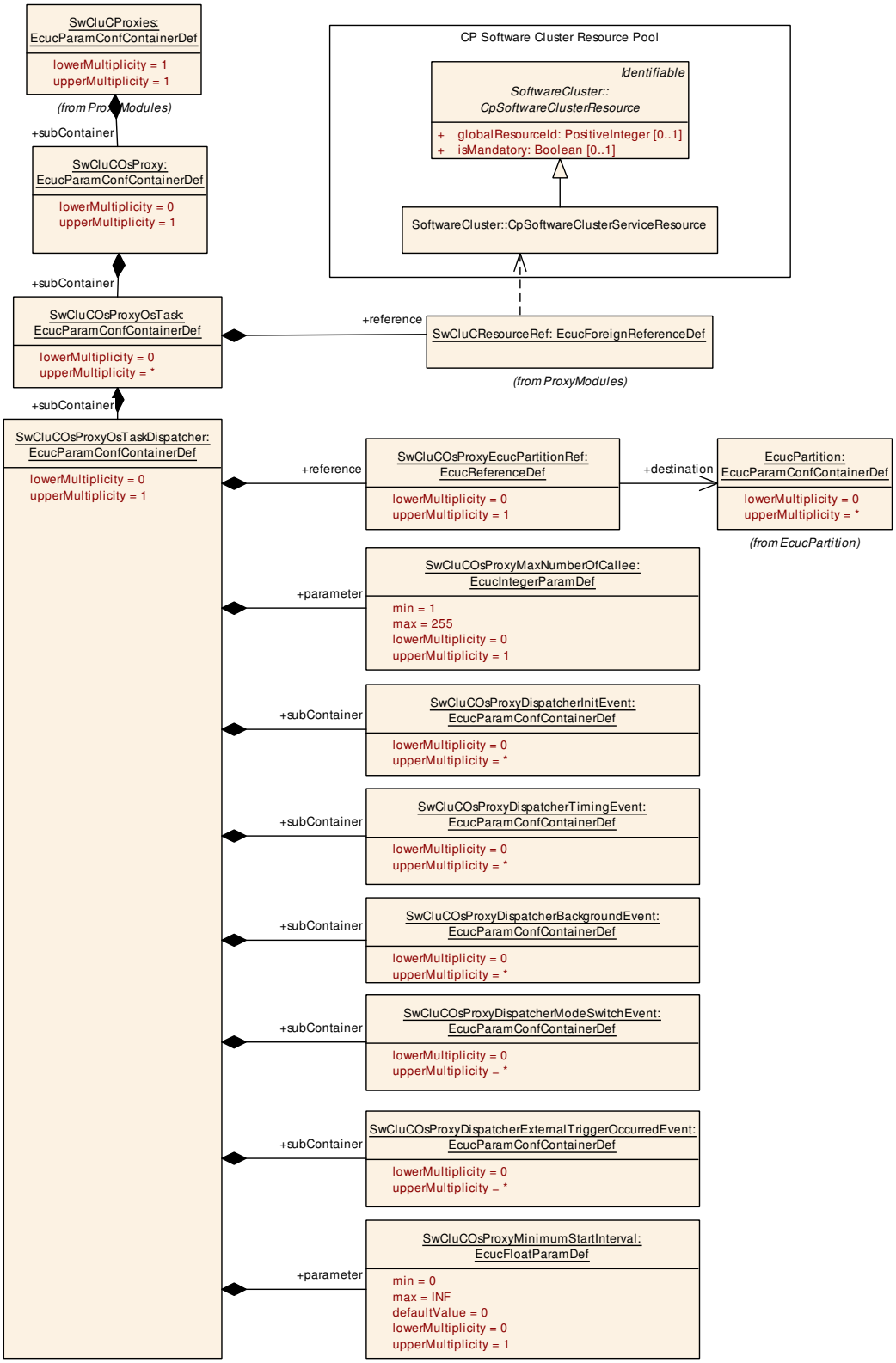

Figure 10.22: Os Proxy Task Dispatcher Parameter

[ECUC_SwCluC_00061] Definition of EcucParamConfContainerDef SwCluCOsProxyOsTaskDispatcher ⌈

Container Name	SwCluCOsProxyOsTaskDispatcher
Parent Container	SwCluCOsProxyOsTask
Upstream mapping defined	–
Description	This container defines a task dispatcher for the owning proxy task in the Os Low Proxy. In the Os High Proxy, this configuration is not used. In the Os Low Proxy, this configuration provides the Dispatcher Runnable. This will cause a call to the SwClu COsProxyOsTask(s) in the Os High Proxy. The Os Low Proxy is able to call one Os ProxyOsTask in each of the MaxNumberOfCallee Os High Proxies. If none of the Sw CluCOsProxyDispatcher<xxx>Event containers is configured it's up to the integrator to model the appropriate RTEEvent.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyMaxNumberOfCallee	0..1	[ECUC_SwCluC_00062]
SwCluCOsProxyMinimumStartInterval	0..1	[ECUC_SwCluC_00070]
SwCluCOsProxyEcucPartitionRef	0..1	[ECUC_SwCluC_00078]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCOsProxyDispatcher BackgroundEvent	0..*	Configures a BackgroundEvent starting the owing Dispatcher Runnable.
SwCluCOsProxyDispatcher ExternalTriggerOccurredEvent	0..*	Configures a ExternalTriggerOccurredEvent starting the owing Dispatcher Runnable.
SwCluCOsProxyDispatcherInit Event	0..*	Configures an InitEvent starting the owing Dispatcher Runnable.
SwCluCOsProxyDispatcherMode SwitchEvent	0..*	Configures a ModeSwitchEvent starting the owing Dispatcher Runnable.
SwCluCOsProxyDispatcherTiming Event	0..*	Configures a TimingEvent starting the owing Dispatcher Runnable.

⌋

[ECUC_SwCluC_00062] Definition of EcucIntegerParamDef SwCluCOsProxyMaxNumberOfCallee ⌈

Parameter Name	SwCluCOsProxyMaxNumberOfCallee	
Parent Container	SwCluCOsProxyOsTaskDispatcher	
Upstream mapping defined	–	
Description	Defines how many Os High Proxy OsProxyOsTask the dispatcher of this Os Low Proxy OsProxyOsTask is able to call at most. In the Os Low Proxy, this configuration parameter is mandatory. In the Os High Proxy, this configuration parameter is not used.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	

▽

△

Pre-Compile only	true

⌋

[ECUC_SwCluC_00070] Definition of EcucFloatParamDef SwCluCOsProxyMinimumStartInterval ⌈

Parameter Name	SwCluCOsProxyMinimumStartInterval	
Parent Container	SwCluCOsProxyOsTaskDispatcher	
Upstream mapping defined	–	
Description	Specifies the time in seconds by which two consecutive starts of an dispatcher Runnable are guaranteed to be separated.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_SwCluC_00078] Definition of EcucReferenceDef SwCluCOsProxyEcucPartitionRef ⌈

Parameter Name	SwCluCOsProxyEcucPartitionRef
Parent Container	SwCluCOsProxyOsTaskDispatcher
Upstream mapping defined	–
Description	If the reference is not provided, the task dispatcher is provided on the EcuPartition to which the OsApplication of the OsTask related to the owning SwCluCOsProxyOsTask belongs. If the reference is provided, the task dispatcher is provided on the referenced Ecu Partition. In this case the task dispatcher is scheduled in the OsTask related to the owning SwCluCOsProxyOsTask but is changing its partition before the task dispatcher schedules proxy tasks in the Application Software Cluster.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

10.2.7.2.6.1 Os Proxy Task Dispatcher Timing Event configuration

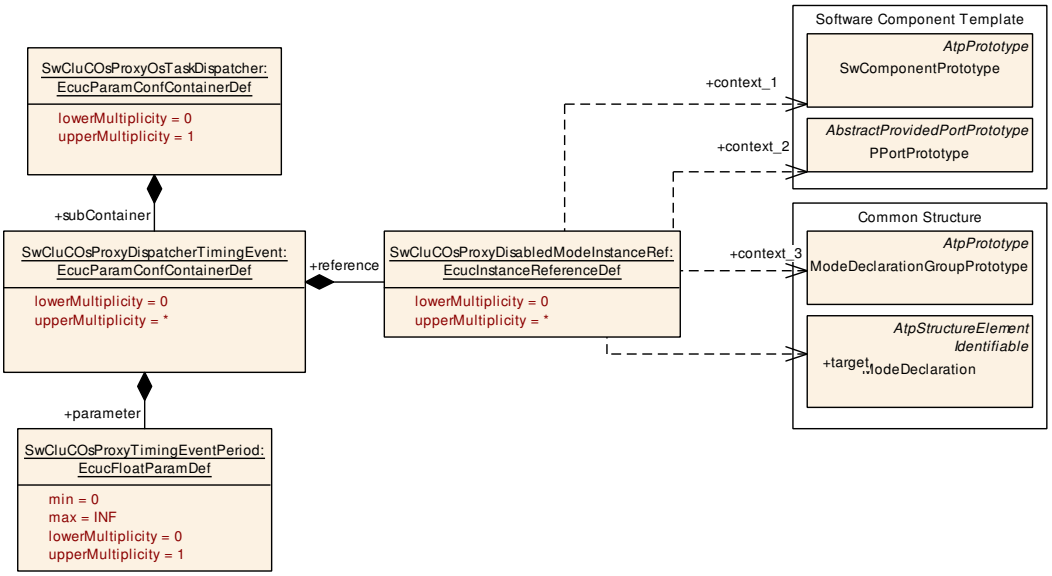

Figure 10.23: Os Proxy Task Dispatcher Timing Event Parameter

[ECUC_SwCluC_00066] Definition of EcucParamConfContainerDef SwCluCOsProxyDispatcherTimingEvent ⌈

Container Name	SwCluCOsProxyDispatcherTimingEvent
Parent Container	SwCluCOsProxyOsTaskDispatcher
Upstream mapping defined	–
Description	Configures a TimingEvent starting the owing Dispatcher Runnable.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyTimingEventPeriod	0..1	[ECUC_SwCluC_00065]
SwCluCOsProxyDisabledModeInstanceRef	0..*	[ECUC_SwCluC_00071]

No Included Containers

⌋

[ECUC_SwCluC_00065] Definition of EcucFloatParamDef SwCluCOsProxyTimingEventPeriod ⌈

Parameter Name	SwCluCOsProxyTimingEventPeriod
Parent Container	SwCluCOsProxyDispatcherTimingEvent
Upstream mapping defined	–
Description	Period of timing event in seconds. The value of this attribute shall be greater than zero.

▽

△

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

⌋

For parameter table [ECUC_SwCluC_00071] SwCluCOsProxyDisabledModeInstanceRef, see definition below container SwCluCOsProxyDispatcherBackgroundEvent.

10.2.7.2.6.2 Os Proxy Task Dispatcher Background Event configuration

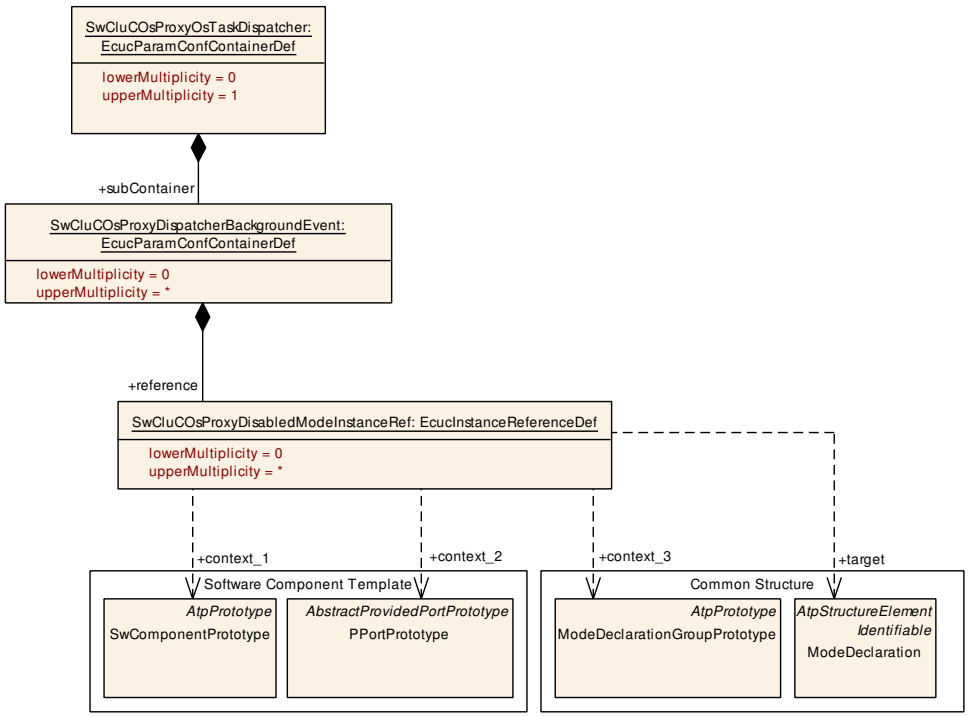

Figure 10.24: Os Proxy Task Dispatcher Background Event Parameter

[ECUC_SwCluC_00067] Definition of EcucParamConfContainerDef SwCluCOsProxyDispatcherBackgroundEvent ⌈

Container Name	SwCluCOsProxyDispatcherBackgroundEvent
Parent Container	SwCluCOsProxyOsTaskDispatcher
Upstream mapping defined	–
Description	Configures a BackgroundEvent starting the owing Dispatcher Runnable.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyDisabledModeInstanceRef	0..*	[ECUC_SwCluC_00071]

No Included Containers

⌋

[ECUC_SwCluC_00071] Definition of EcucInstanceReferenceDef SwCluCOsProxyDisabledModeInstanceRef ⌈

Parameter Name	SwCluCOsProxyDisabledModeInstanceRef
Parent Container	SwCluCOsProxyDispatcherBackgroundEvent, SwCluCOsProxyDispatcherExternalTriggerOccurredEvent, SwCluCOsProxyDispatcherModeSwitchEvent, SwCluCOsProxyDispatcherTimingEvent
Upstream mapping defined	–
Description	Reference to the Mode instance in a PPortPrototype that disable the Event.
Multiplicity	0..*
Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Pre-Compile only	true

⌋

10.2.7.2.6.3 Os Proxy Task Dispatcher Init Event configuration

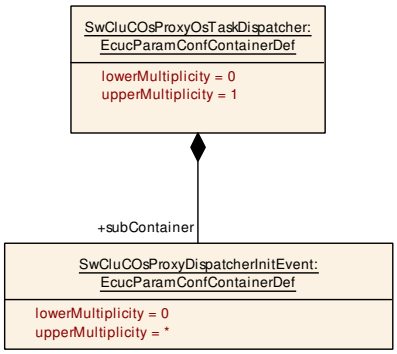

Figure 10.25: Os Proxy Task Dispatcher Init Event Parameter

[ECUC_SwCluC_00153] Definition of EcucParamConfContainerDef SwCluCOsProxyDispatcherInitEvent ⌈

Container Name	SwCluCOsProxyDispatcherInitEvent
Parent Container	SwCluCOsProxyOsTaskDispatcher
Upstream mapping defined	–
Description	Configures an InitEvent starting the owing Dispatcher Runnable.
Multiplicity	0..*

▽

△

Pre-Compile only	true
Configuration Parameters	

No Included Parameters

No Included Containers

⌋

10.2.7.2.6.4 Os Proxy Task Dispatcher Mode Switch Event configuration

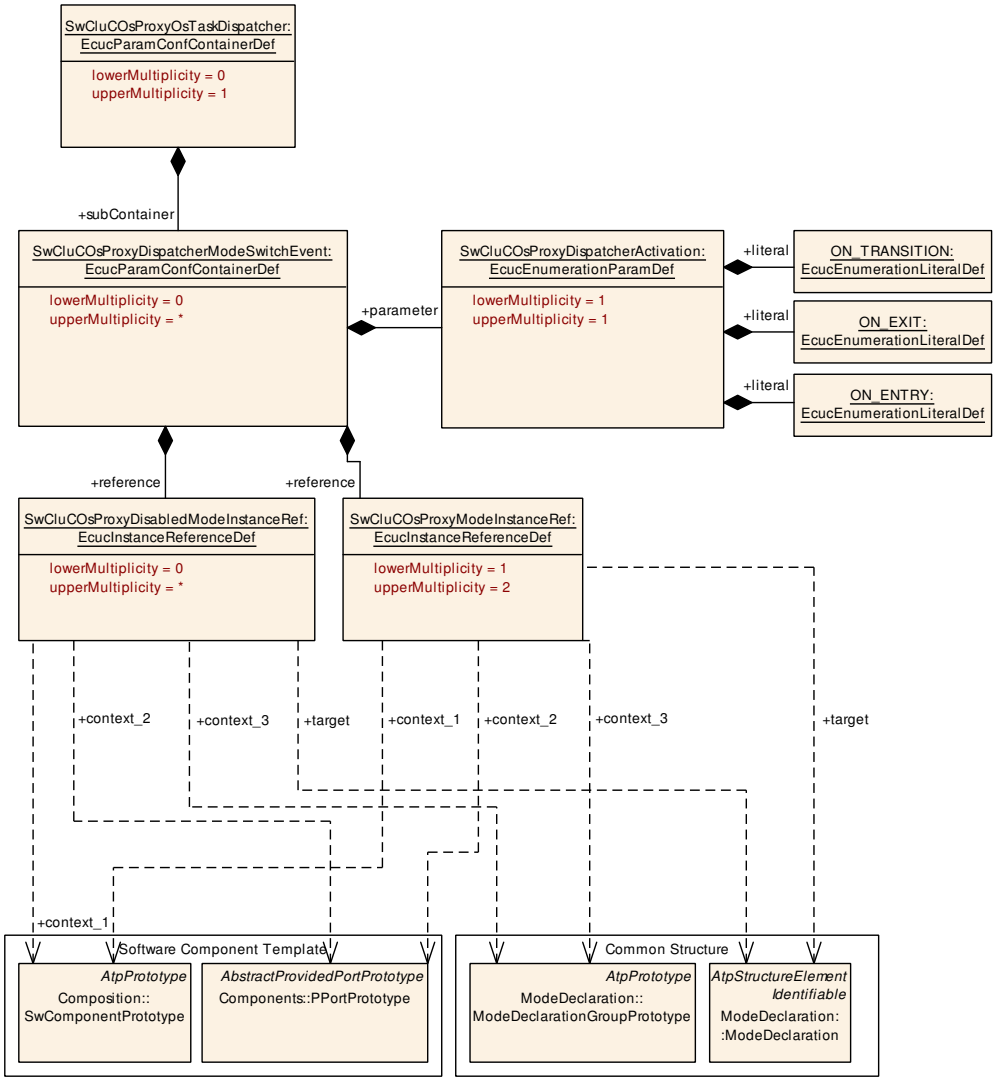

Figure 10.26: Os Proxy Task Dispatcher Mode Switch Event Parameter

[ECUC_SwCluC_00068] Definition of EcucParamConfContainerDef SwCluCOs ProxyDispatcherModeSwitchEvent ⌈

Container Name	SwCluCOsProxyDispatcherModeSwitchEvent
Parent Container	SwCluCOsProxyOsTaskDispatcher
Upstream mapping defined	–
Description	Configures a ModeSwitchEvent starting the owing Dispatcher Runnable.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCOsProxyDispatcherActivation	1	[ECUC_SwCluC_00072]
SwCluCOsProxyDisabledModeInstanceRef	0..*	[ECUC_SwCluC_00071]
SwCluCOsProxyModeInstanceRef	1..2	[ECUC_SwCluC_00073]

No Included Containers

⌋

[ECUC_SwCluC_00072] Definition of EcucEnumerationParamDef SwCluCOsProxyDispatcherActivation ⌈

Parameter Name	SwCluCOsProxyDispatcherActivation	
Parent Container	SwCluCOsProxyDispatcherModeSwitchEvent	
Upstream mapping defined	–	
Description	Specifies if the event is activated on entering or exiting the referenced Mode instance.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ON_ENTRY	On entering the referred mode.
	ON_EXIT	On exiting the referred mode.
	ON_TRANSITION	On transition of the 1st referred mode to the 2nd referred mode.
Pre-Compile only	true	

⌋

For parameter table [ECUC_SwCluC_00071] SwCluCOsProxyDisabledModeInstanceRef, see definition below container SwCluCOsProxyDispatcherBackgroundEvent.

[ECUC_SwCluC_00073] Definition of EcucInstanceReferenceDef SwCluCOsProxyModeInstanceRef ⌈

Parameter Name	SwCluCOsProxyModeInstanceRef
Parent Container	SwCluCOsProxyDispatcherModeSwitchEvent
Upstream mapping defined	–
Description	Reference to one or two Mode instances in a PPortPrototype that that initiate the mode switch.
Multiplicity	1..2

▽

△

Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Pre-Compile only	true

⌋

10.2.7.2.6.5 Os Proxy Task Dispatcher External Trigger Occurred Event configuration

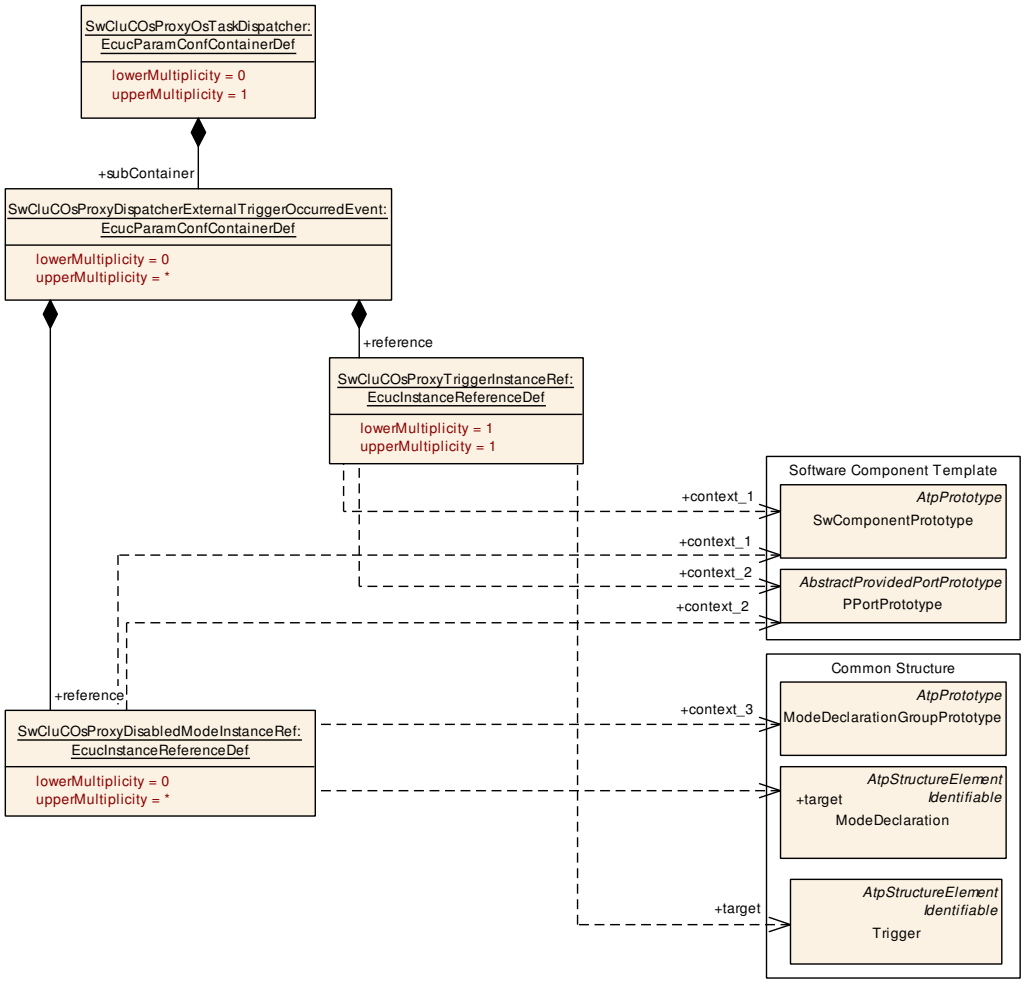

Figure 10.27: Os Proxy Task Dispatcher External Trigger Occurred Event Parameter

[ECUC_SwCluC_00069] Definition of EcucParamConfContainerDef SwCluCOsProxyDispatcherExternalTriggerOccurredEvent ⌈

Container Name	SwCluCOsProxyDispatcherExternalTriggerOccurredEvent
Parent Container	SwCluCOsProxyOsTaskDispatcher
Upstream mapping defined	–
Description	Configures a ExternalTriggerOccurredEvent starting the owing Dispatcher Runnable.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCOsProxyDisabledModeInstanceRef	0..*	[ECUC_SwCluC_00071]
SwCluCOsProxyTriggerInstanceRef	1	[ECUC_SwCluC_00074]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00071] SwCluCOsProxyDisabledModeInstanceRef, see definition below container SwCluCOsProxyDispatcherBackgroundEvent.

[ECUC_SwCluC_00074] Definition of EcucInstanceReferenceDef SwCluCOsProxyTriggerInstanceRef ⌈

Parameter Name	SwCluCOsProxyTriggerInstanceRef
Parent Container	SwCluCOsProxyDispatcherExternalTriggerOccurredEvent
Upstream mapping defined	–
Description	Reference to the Trigger instances in a PPortPrototype that raise the trigger.
Multiplicity	1
Type	Instance reference to TRIGGER context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE
Pre-Compile only	true

⌋

10.2.7.2.7 Os Proxy Function Dispatcher configuration

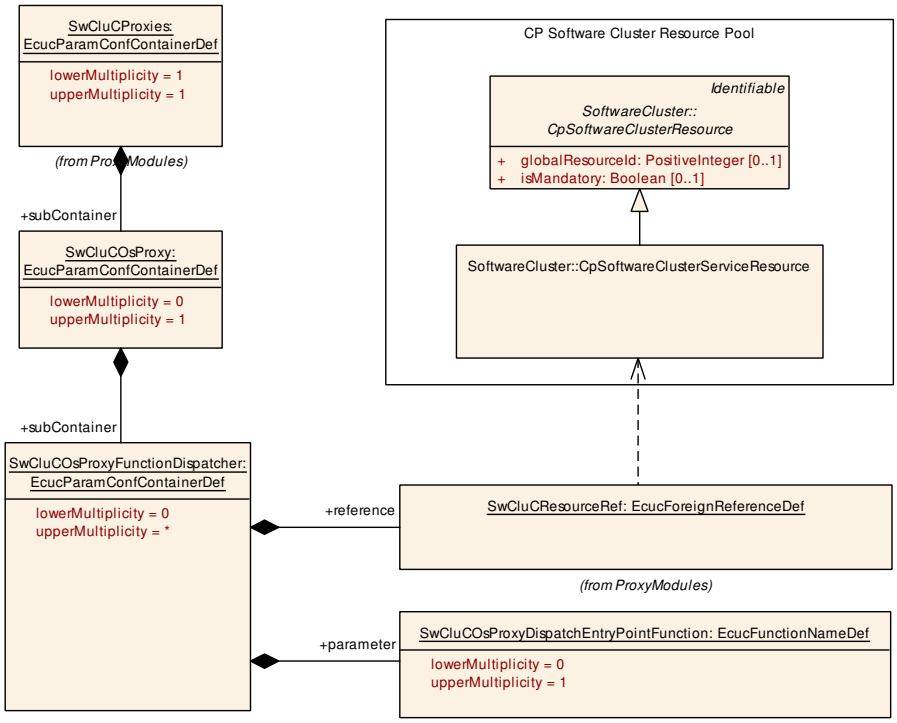

Figure 10.28: Os Proxy Function Dispatcher Parameter

[ECUC_SwCluC_00063] Definition of EcucParamConfContainerDef SwCluCOs ProxyFunctionDispatcher ⌈

Container Name	SwCluCOsProxyFunctionDispatcher
Parent Container	SwCluCOsProxy
Upstream mapping defined	–
Description	This container configures a function dispatcher. In the Os High Proxy, this configuration creates a Dispatch Entry Point in the Binary Manifest, which can be called by the Os Low Proxy. In the Os Low Proxy, this configuration provides the Dispatcher Function (and according resource Entry in the Binary Manifest), which is able to call the Max Number OfCallee Dispatch Entry Points of Os High Proxies.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCOsProxyDispatchEntryPointFunction	0..1	[ECUC_SwCluC_00064]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00064] Definition of EcucFunctionNameDef SwCluCOsProxy DispatchEntryPointFunction ⌈

Parameter Name	SwCluCOsProxyDispatchEntryPointFunction
Parent Container	SwCluCOsProxyFunctionDispatcher
Upstream mapping defined	–
Description	Name of the function, which is put into the Binary Manifest as Dispatch Entry Point. In the Os High Proxy, this configuration parameter is mandatory. In the Os Low Proxy, this configuration parameter is not used.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02237] `SwCluCOsProxyFunctionDispatcher` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_FNC_DISPATCHER`

Upstream requirements: SRS_SwCluC_00214, SRS_BSW_00167

⌈The `SwCluCOsProxyFunctionDispatcher.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_OS_FNC_DISPATCHER`.⌋

10.2.7.3 Com Proxy

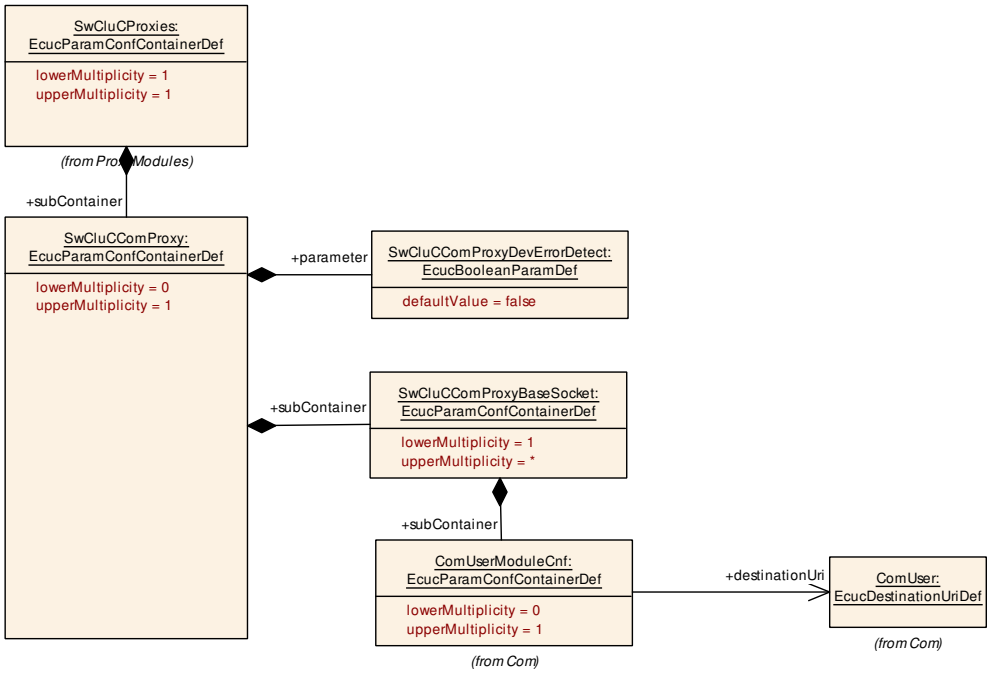

Figure 10.29: Com Proxy Parameter

[ECUC_SwCluC_00127] Definition of EcucParamConfContainerDef SwCluCCom Proxy ⌈

Container Name	SwCluCComProxy
Parent Container	SwCluCProxies
Upstream mapping defined	–
Description	Com Proxy specific parameters.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCComProxyDevErrorDetect	1	[ECUC_SwCluC_00128]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCComProxyBaseSocket	1..*	This container configures how many EcucPartitions specific API links are required for the Com Proxy and to which cluster resource the API set belongs. In the Com Low Proxy all offered API sets (incl callbacks) needs to be configured. In the Com High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the Ecuc Partition on which the API set is provided. The ComUserModuleCnf is only applicable in the Host Software Cluster and shall not be configured in the scope of a Application Software Cluster. One ComUserModuleCnf instance can either server for ComSignals, ComSignalGroups, and ComIPdus on multiple EcucPartitions or exactly one EcucPartition. How many ComUser instances an Rte in an related Application Software Cluster requires is vendor specific.

⌋

[ECUC_SwCluC_00128] Definition of EcucBooleanParamDef SwCluCComProxy DevErrorDetect ⌈

Parameter Name	SwCluCComProxyDevErrorDetect
Parent Container	SwCluCComProxy
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

⌋

10.2.7.3.1 Com Proxy Base Socket

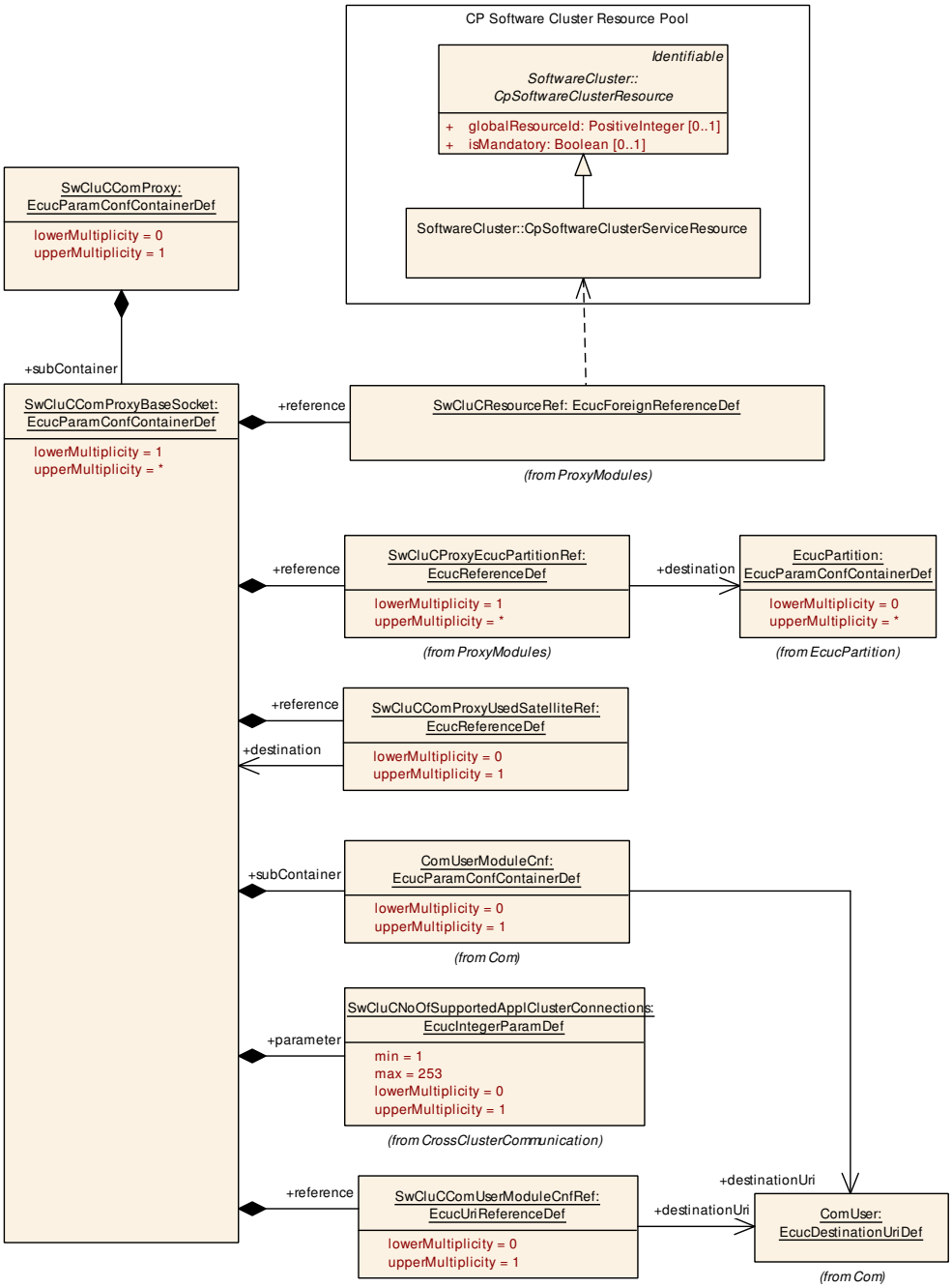

Figure 10.30: Com Proxy Base Socket

[ECUC_SwCluC_00131] Definition of EcucParamConfContainerDef SwCluCCom ProxyBaseSocket ⌈

Container Name	SwCluCComProxyBaseSocket
Parent Container	SwCluCComProxy
Upstream mapping defined	–
Description	This container configures how many EcucPartitions specific API links are required for the Com Proxy and to which cluster resource the API set belongs. In the Com Low Proxy all offered API sets (incl callbacks) needs to be configured. In the Com High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the API set is provided. The ComUserModuleCnf is only applicable in the Host Software Cluster and shall not be configured in the scope of a Application Software Cluster. One ComUserModuleCnf instance can either server for ComSignals, ComSignalGroups, and ComIPdus on multiple EcucPartitions or exactly one EcucPartition. How many ComUser instances an Rte in an related Application Software Cluster requires is vendor specific.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCComProxyUsedSatelliteRef	0..1	[ECUC_SwCluC_00132]
SwCluCComUserModuleCnfRef	0..1	[ECUC_SwCluC_00167]
SwCluCProxyEcucPartitionRef	1..*	[ECUC_SwCluC_00147]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

Included Containers		
Container Name	**Multiplicity**	**Description**
ComUserModuleCnf	0..1	Contains the configuration parameters of the Com user module.

⌋

[ECUC_SwCluC_00166] Definition of EcucIntegerParamDef SwCluCNoOfSupportedApplClusterConnections ⌈

Parameter Name	SwCluCNoOfSupportedApplClusterConnections	
Parent Container	SwCluCComProxyBaseSocket, SwCluCDcmProxyBaseSocket, SwCluCDemProxyBaseSocket, SwCluCFiMProxyBaseSocket, SwCluCNvMBaseSocket, SwCluCOsProxyOsBaseSocket, SwCluCXccBaseSocket	
Upstream mapping defined	–	
Description	This parameter configures how many Application Software Clusters can connect at most to this Socket. This parameter is only applicable in the configuration of an Low Proxy. If the parameter is not given in the Low Proxy the according base socket supports at most one connection to an Application Software Cluster.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 253	
Default value	–	

▽

△

Pre-Compile only	true

⌋

[ECUC_SwCluC_00132] Definition of EcucReferenceDef SwCluCComProxyUsedSatelliteRef ⌈

Parameter Name	SwCluCComProxyUsedSatelliteRef
Parent Container	SwCluCComProxyBaseSocket
Upstream mapping defined	–
Description	Reference to the SwCluCComBaseSocket which has access to a satellite of the Com. The owning SwCluCComBaseSocket uses the Com satellite of the referenced SwCluCComBaseSocket.
Multiplicity	0..1
Type	Reference to SwCluCComProxyBaseSocket
Pre-Compile only	true

⌋

[ECUC_SwCluC_00167] Definition of EcucUriReferenceDef SwCluCComUserModuleCnfRef ⌈

Parameter Name	SwCluCComUserModuleCnfRef
Parent Container	SwCluCComProxyBaseSocket
Upstream mapping defined	–
Description	Reference to the ComUserModuleCnf of Rte. This reference shall only be used in the configuration of Application Software Clusters.
Multiplicity	0..1
Type	Reference to destinationUri ComUserUriDefSet/ComUser
Pre-Compile only	true

⌋

[ECUC_SwCluC_00147] Definition of EcucReferenceDef SwCluCProxyEcucPartitionRef ⌈

Parameter Name	SwCluCProxyEcucPartitionRef
Parent Container	SwCluCComProxyBaseSocket
Upstream mapping defined	–
Description	Reference to the EcucPartition.
Multiplicity	1..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02664] `SwCluCComProxyBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_COM_BASE_SOCKET`

Upstream requirements: SRS_SwCluC_00211, SRS_BSW_00167

⌈The `SwCluCComProxyBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_COM_BASE_SOCKET`.⌋

10.2.7.3.2 Com User Module Configuration

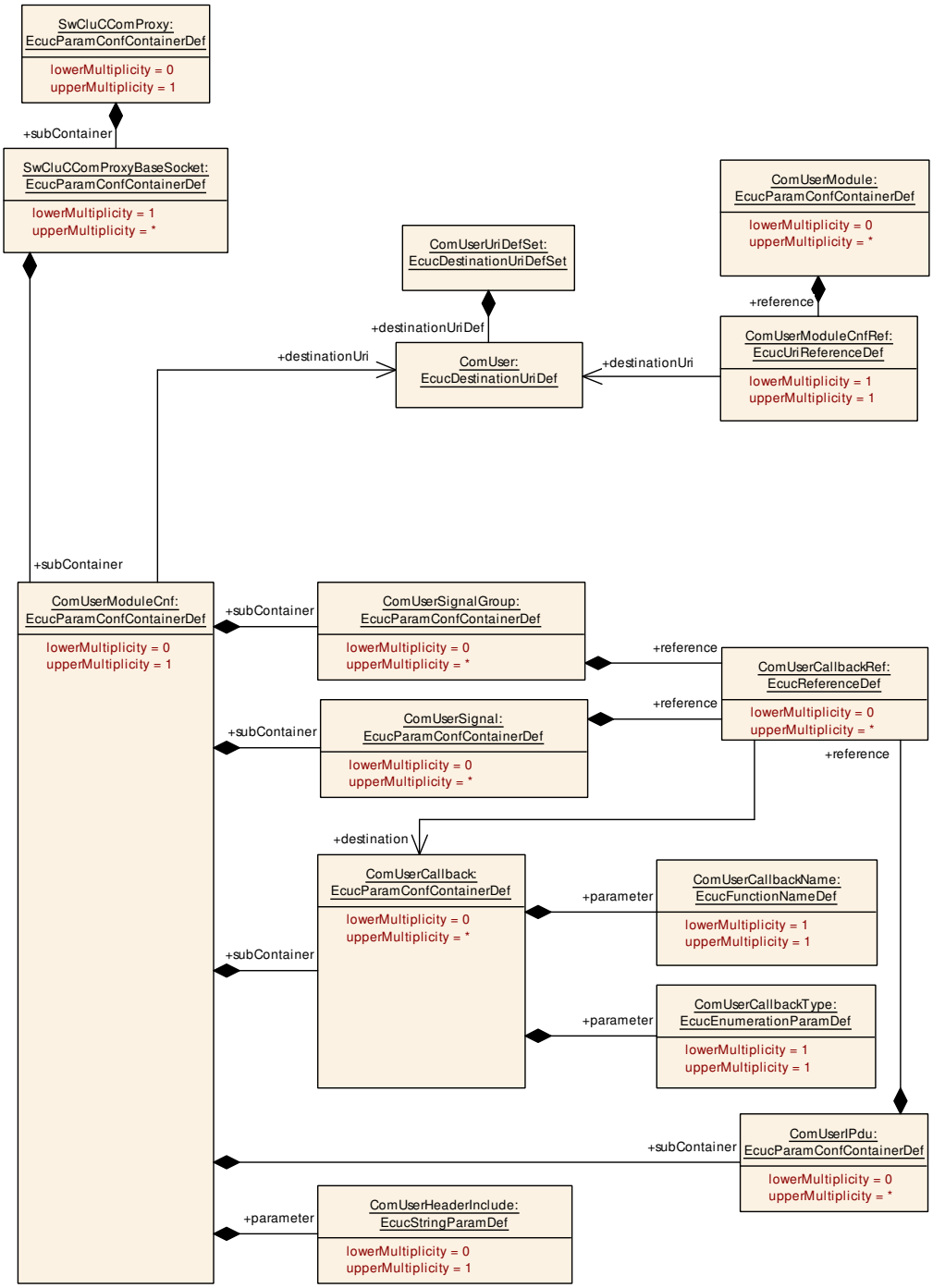

Figure 10.31: Com user module configuration

[ECUC_Com_10030] Definition of EcucParamConfContainerDef ComUserModule Cnf

Container Name	ComUserModuleCnf
Parent Container	RteComUser, SwCluCComProxyBaseSocket
Parent Destination Uri Definition	ComUser
Upstream mapping defined	–
Description	Contains the configuration parameters of the Com user module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
ComUserHeaderInclude	0..1	[ECUC_Com_10032]

Included Containers		
Container Name	Multiplicity	Description
ComUserCallback	0..*	This container defines a Com callback function for ComSignals, ComSignalGroups, and ComIPdus.
ComUserIPdu	0..*	Contains the configuration parameters for the ComIPdu with Large Data Path support inside a Com user module.
ComUserSignal	0..*	Contains the configuration parameters of a signal inside a Com user module. Please note that it is valid to define ComUser Signal without any callback function.
ComUserSignalGroup	0..*	Contains the configuration parameters of a signal group inside a Com user module. Please note that it is valid to define ComUser SignalGroup without any callback function.

[ECUC_Com_10032] Definition of EcucStringParamDef ComUserHeaderInclude

Parameter Name	ComUserHeaderInclude
Parent Container	ComUserModuleCnf
Upstream mapping defined	–
Description	Defines the header file where the Com user provides the function declarations for configured callbacks.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

10.2.7.3.3 Com User IPdus

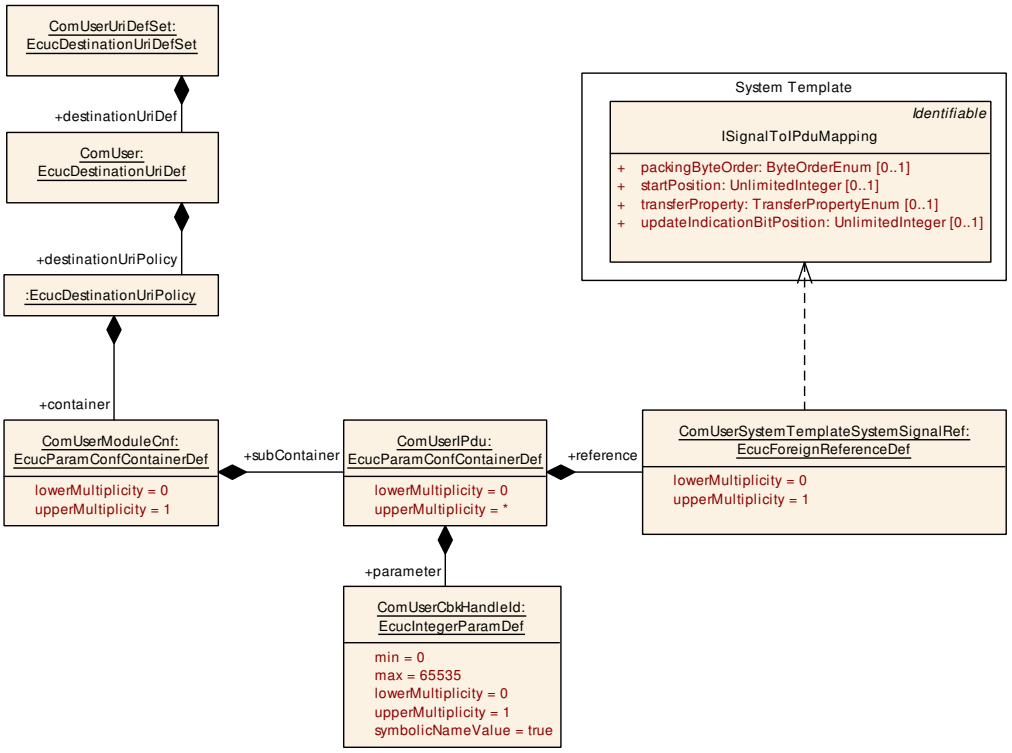

Figure 10.32: Com user IPdus

[ECUC_Com_10048] Definition of EcucParamConfContainerDef ComUserIPdu

Container Name	ComUserIPdu	
Parent Container	ComUserModuleCnf	
Upstream mapping defined	–	
Description	Contains the configuration parameters for the ComIPdu with Large Data Path support inside a Com user module.	
Multiplicity	0..*	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComUserCbkHandleId	0..1	[ECUC_Com_10033]
ComUserCallbackRef	0..*	[ECUC_Com_10035]
ComUserSystemTemplateSystemSignalRef	0..1	[ECUC_Com_10025]

No Included Containers

[ECUC_Com_10033] Definition of EcucIntegerParamDef ComUserCbkHandleId

Parameter Name	ComUserCbkHandleId	
Parent Container	ComUserIPdu, ComUserSignal, ComUserSignalGroup	
Upstream mapping defined	–	
Description	The numerical value used as the Com user callback handle Id. This ID identifies signals and signal groups in the COM callbacks using ComUserCbk HandleId parameter respectively.	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Com_10035] Definition of EcucReferenceDef ComUserCallbackRef

Parameter Name	ComUserCallbackRef	
Parent Container	ComUserIPdu, ComUserSignal, ComUserSignalGroup	
Upstream mapping defined	–	
Description	Reference(s) to all callback(s) of this signal or signal group or IPdu.	
Multiplicity	0..*	
Type	Reference to ComUserCallback	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	-	Selectable

[ECUC_Com_10025] Definition of EcucForeignReferenceDef ComUserSystemTemplateSystemSignalRef

Parameter Name	ComUserSystemTemplateSystemSignalRef	
Parent Container	ComUserIPdu, ComUserSignal	
Upstream mapping defined	–	
Description	Reference to the ISignalToIPduMapping that contains a reference to the ISignal (System Template) which this ComUserSignal (or ComUserGroupSignal) represents.	
Multiplicity	0..1	
Type	Foreign reference to I-SIGNAL-TO-I-PDU-MAPPING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

10.2.7.3.4 Com User Signals and Signal Groups

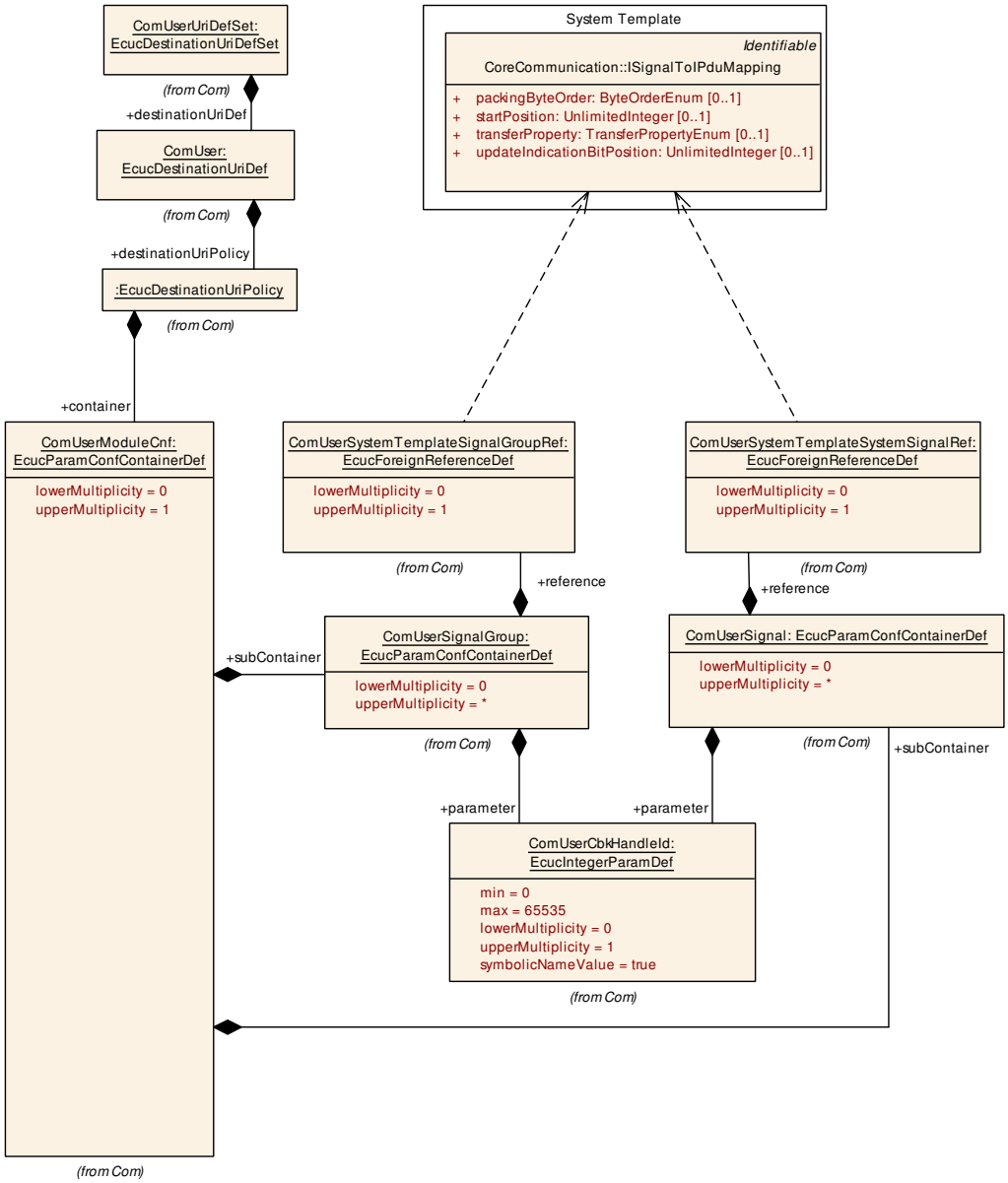

Figure 10.33: Com user signals and signal groups

10.2.7.3.4.1 Com User Signals

[ECUC_Com_10028] Definition of EcucParamConfContainerDef ComUserSignal

Container Name	ComUserSignal
Parent Container	ComUserModuleCnf
Upstream mapping defined	–

▽

△

Description	Contains the configuration parameters of a signal inside a Com user module. Please note that it is valid to define ComUserSignal without any callback function.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComUserCbkHandleId	0..1	[ECUC_Com_10033]
ComUserCallbackRef	0..*	[ECUC_Com_10035]
ComUserSystemTemplateSystemSignalRef	0..1	[ECUC_Com_10025]

No Included Containers

For parameter table [ECUC_Com_10033] ComUserCbkHandleId, see definition below container ComUserIPdu.

For parameter table [ECUC_Com_10035] ComUserCallbackRef, see definition below container ComUserIPdu.

For parameter table [ECUC_Com_10025] ComUserSystemTemplateSystemSignalRef, see definition below container ComUserIPdu.

10.2.7.3.4.2 Com User Signal Groups

[ECUC_Com_10027] Definition of EcucParamConfContainerDef ComUserSignalGroup

Container Name	ComUserSignalGroup	
Parent Container	ComUserModuleCnf	
Upstream mapping defined	–	
Description	Contains the configuration parameters of a signal group inside a Com user module. Please note that it is valid to define ComUserSignalGroup without any callback function.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComUserCbkHandleId	0..1	[ECUC_Com_10033]
ComUserCallbackRef	0..*	[ECUC_Com_10035]
ComUserSystemTemplateSignalGroupRef	0..1	[ECUC_Com_10026]

No Included Containers

For parameter table [ECUC_Com_10033] ComUserCbkHandleId, see definition below container ComUserIPdu.

For parameter table [ECUC_Com_10035] ComUserCallbackRef, see definition below container ComUserIPdu.

[ECUC_Com_10026] Definition of EcucForeignReferenceDef ComUserSystemTemplateSignalGroupRef

Parameter Name	ComUserSystemTemplateSignalGroupRef	
Parent Container	ComUserSignalGroup	
Upstream mapping defined	–	
Description	Reference to the ISignalToIPduMapping that contains a reference to the ISignalGroup (SystemTemplate) which this ComUserSignalGroup represents.	
Multiplicity	0..1	
Type	Foreign reference to I-SIGNAL-TO-I-PDU-MAPPING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

10.2.7.3.5 Com User callbacks

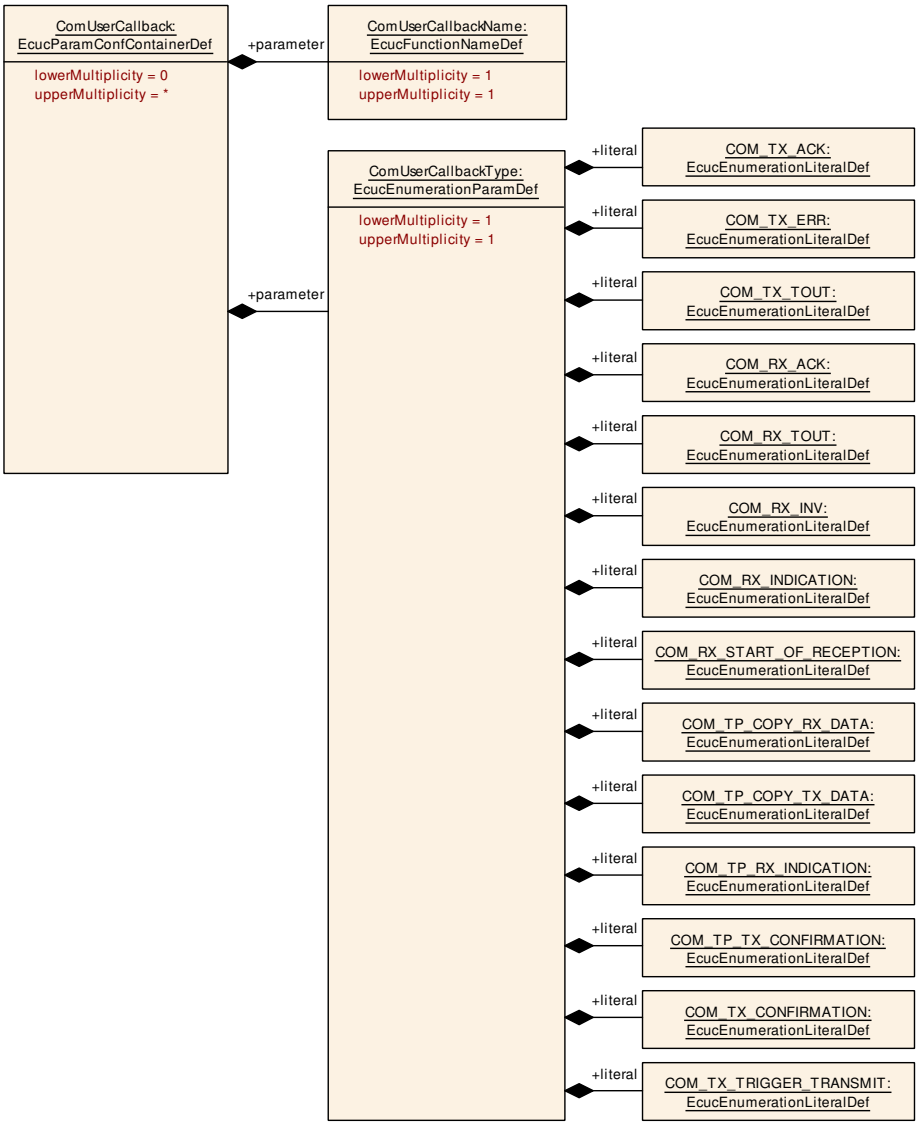

Figure 10.34: Com User callbacks

[ECUC_Com_10038] Definition of EcucParamConfContainerDef ComUserCallback

Container Name	ComUserCallback	
Parent Container	ComUserModuleCnf	
Upstream mapping defined	–	
Description	This container defines a Com callback function for ComSignals, ComSignalGroups, and ComIPdus.	
Multiplicity	0..*	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComUserCallbackName	1	[ECUC_Com_10036]
ComUserCallbackType	1	[ECUC_Com_10034]

No Included Containers

[ECUC_Com_10036] Definition of EcucFunctionNameDef ComUserCallback Name

Parameter Name	ComUserCallbackName
Parent Container	ComUserCallback
Upstream mapping defined	–
Description	The name of the callback function to be called.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Com_10034] Definition of EcucEnumerationParamDef ComUserCallback Type

Parameter Name	ComUserCallbackType	
Parent Container	ComUserCallback	
Upstream mapping defined	–	
Description	The type of the Com callback	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	COM_RX_ACK	Com_CbkRxAck callback which is called immediately after the message has been stored in the receiving message object. This type of callback represents a ComNotification.
	COM_RX_INDICATION	ComCbkRxIndication Large Data Path callback indicates a received PDU from a lower layer communication interface module.
	COM_RX_INV	Com_CbkInv callback which is called after reception of an invalid signal or signal group respectively. This type of callback represents a ComInvalidNotification.
	COM_RX_START_OF_RECEPTION	ComCbkStartOfReception Large Data Path callback called at the start of receiving an N-SDU.
	COM_RX_TOUT	Com_CbkRxTOut callback which is called immediately after a message reception error has been detected by the deadline monitoring mechanism. This type of callback represents a ComTimeoutNotification.
	COM_TP_COPY_RX_DATA	ComCbkCopyRxData Large Data Path callback to provide the received data of an I-PDU segment (N-PDU) to the upper layer.



△

	COM_TP_COPY_TX_DATA	ComCbkCopyTxData Large Data Path callback to acquire the transmit data of an I-PDU segment.
	COM_TP_RX_INDICATION	ComCbkTpRxIndication Large Data Path callback called after an I-PDU has been received via the TP API.
	COM_TP_TX_CONFIRMATION	ComCbkTpTxConfirmation Large Data Path callback called after a Signal has been transmitted via the TP-API on its network.
	COM_TX_ACK	Com_CbkTxAck callback which is called immediately after successful transmission of the I-PDU containing the message. This type of callback represents a ComNotification.
	COM_TX_CONFIRMATION	ComCbkTxConfirmation Large Data Path callback which is called when the lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
	COM_TX_ERR	Com_CbkTxErr callback which is called in case the transmission is not possible because the corresponding I-PDU group is stopped. This type of callback represents a ComErrorNotification.
	COM_TX_TOUT	Com_CbkTxTOut callback which is called immediately after a message transmission error has been detected by the deadline monitoring mechanism. This type of callback represents a ComTimeoutNotification.
	COM_TX_TRIGGER_TRANSMIT	Large Data Path ComCbkTriggerTransmit callback to acquire transmit data for the lower layer communication interface module.
Pre-Compile only	true	

10.2.7.4 Dcm Proxy

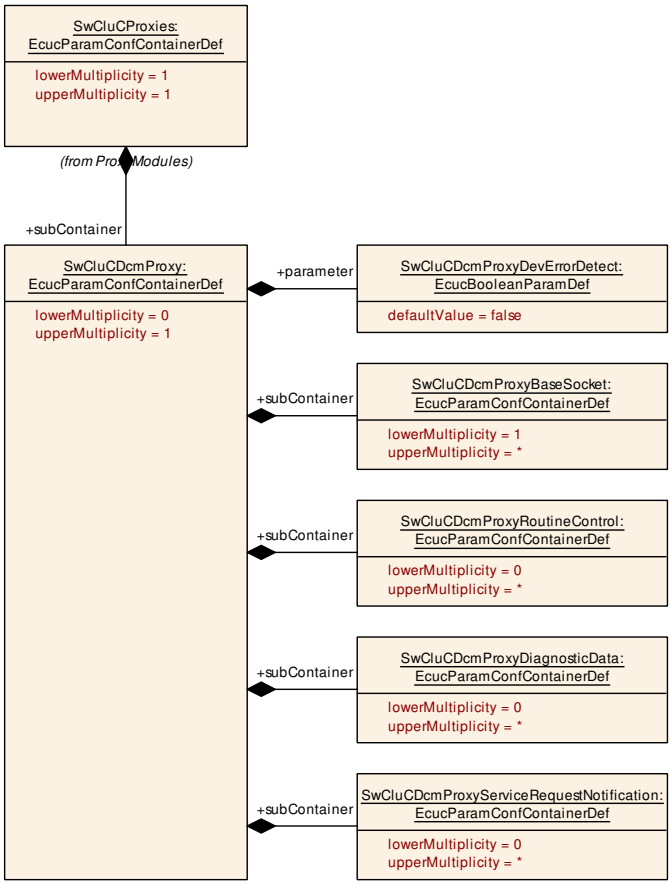

Figure 10.35: Dcm Proxy Parameter

[ECUC_SwCluC_00118] Definition of EcucParamConfContainerDef SwCluCDcm Proxy ⌈

Container Name	SwCluCDcmProxy
Parent Container	SwCluCProxies
Upstream mapping defined	–
Description	Dcm Proxy specific parameters.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCDcmProxyDevErrorDetect	1	[ECUC_SwCluC_00137]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCDcmProxyBaseSocket	1..*	This container configures how many EcucPartitions specific API links are required for the Dcm Proxy and to which cluster resource the API set belongs. In the Dcm Low Proxy all offered API sets needs to be configured. In the Dcm High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the Ecuc Partition on which the API set is provided.
SwCluCDcmProxyDiagnosticData	0..*	Configures a Diagnostic Data proxy for a DID or PID or IOControl.
SwCluCDcmProxyRoutineControl	0..*	Configures a Routine Control proxy.
SwCluCDcmProxyServiceRequestNotification	0..*	Configures a Service Request Notification proxy.

⌋

[ECUC_SwCluC_00137] Definition of EcucBooleanParamDef SwCluCDcmProxyDevErrorDetect ⌈

Parameter Name	SwCluCDcmProxyDevErrorDetect
Parent Container	SwCluCDcmProxy
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

⌋

10.2.7.4.1 Dcm Base Socket

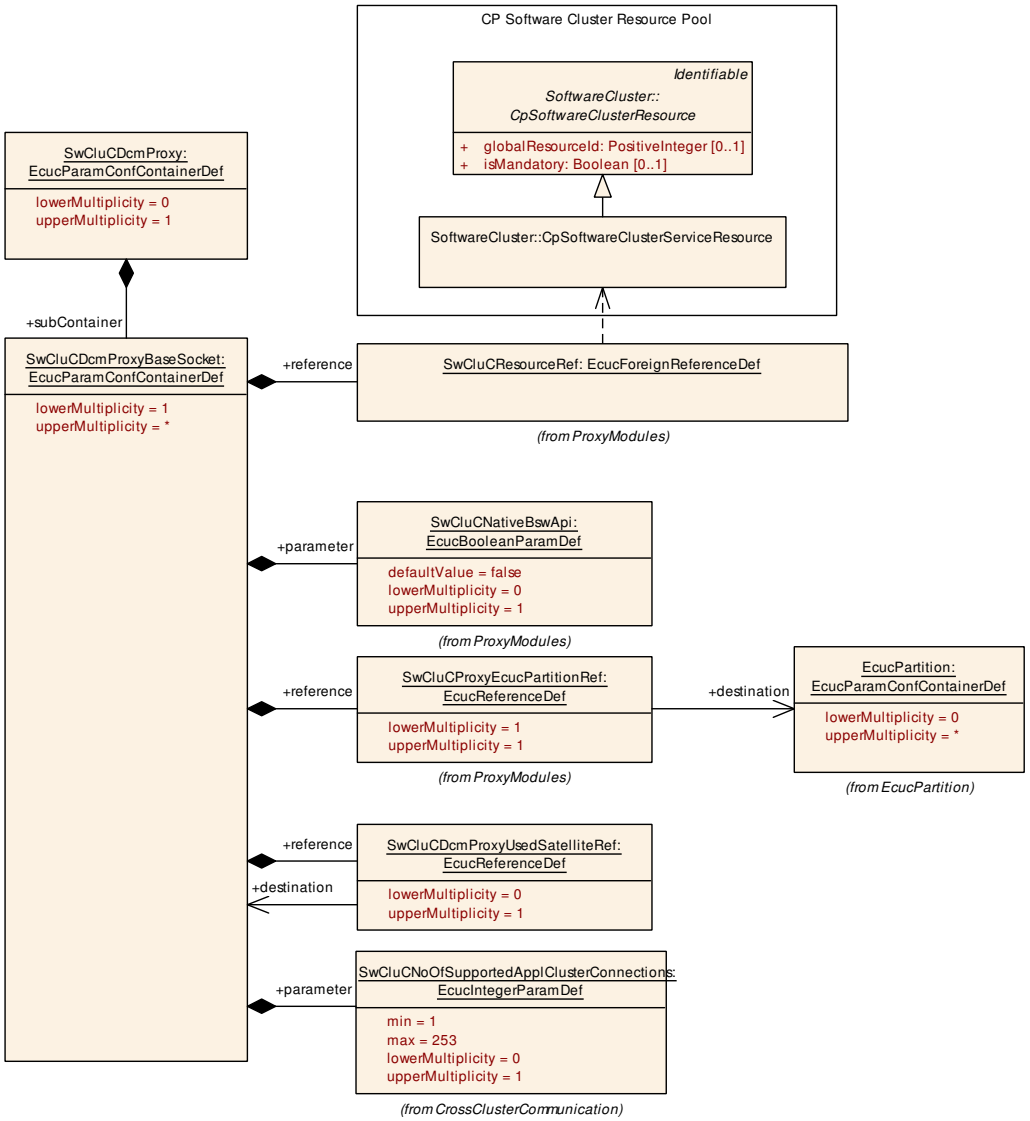

Figure 10.36: Dcm Proxy Base Socket

[ECUC_SwCluC_00120] Definition of EcucParamConfContainerDef SwCluCDcm ProxyBaseSocket ⌈

Container Name	SwCluCDcmProxyBaseSocket
Parent Container	SwCluCDcmProxy
Upstream mapping defined	–

▽

△

Description	This container configures how many EcucPartitions specific API links are required for the Dcm Proxy and to which cluster resource the API set belongs. In the Dcm Low Proxy all offered API sets needs to be configured. In the Dcm High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the API set is provided.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNativeBswApi	0..1	[ECUC_SwCluC_00058]
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCDcmProxyUsedSatelliteRef	0..1	[ECUC_SwCluC_00121]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00058] Definition of EcucBooleanParamDef SwCluCNativeBswApi ⌈

Parameter Name	SwCluCNativeBswApi
Parent Container	SwCluCDcmProxyBaseSocket, SwCluCDemProxyBaseSocket, SwCluCFiMProxyBaseSocket
Upstream mapping defined	–
Description	Defines if the native C-API without pre or suffixes is offered in the Application Software Cluster on this EcucPartition.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00166] SwCluCNoOfSupportedApplClusterConnections, see definition below container SwCluCComProxyBaseSocket.

[ECUC_SwCluC_00121] Definition of EcucReferenceDef SwCluCDcmProxyUsedSatelliteRef ⌈

Parameter Name	SwCluCDcmProxyUsedSatelliteRef
Parent Container	SwCluCDcmProxyBaseSocket
Upstream mapping defined	–

▽

△

Description	Reference to the SwCluCDemBaseSocket which has access to a satellite of the Dcm. The owning SwCluCDemBaseSocket uses the Dcm satellite of the referenced SwCluCDemBaseSocket.
Multiplicity	0..1
Type	Reference to SwCluCDcmProxyBaseSocket
Pre-Compile only	true

⌋

[ECUC_SwCluC_00059] Definition of EcucReferenceDef SwCluCProxyEcucPartitionRef ⌈

Parameter Name	SwCluCProxyEcucPartitionRef
Parent Container	SwCluCDcmProxyBaseSocket, SwCluCDcmProxyDiagnosticData, SwCluCDcmProxyRoutineControl, SwCluCDcmProxyServiceRequestNotification, SwCluCDemProxyBaseSocket, SwCluCDemProxyDiagnosticEvent, SwCluCFiMProxyBaseSocket, SwCluCFiMProxyFID, SwCluCNvMBaseSocket
Upstream mapping defined	–
Description	Reference to the EcucPartition.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02752] `SwCluCDemProxyBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DCM_BASE_SOCKET`

Upstream requirements: SRS_SwCluC_00208, SRS_BSW_00167

⌈The `SwCluCDcmProxyBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DCM_BASE_SOCKET`.⌋

10.2.7.4.2 Dcm Proxy Diagnostic Data

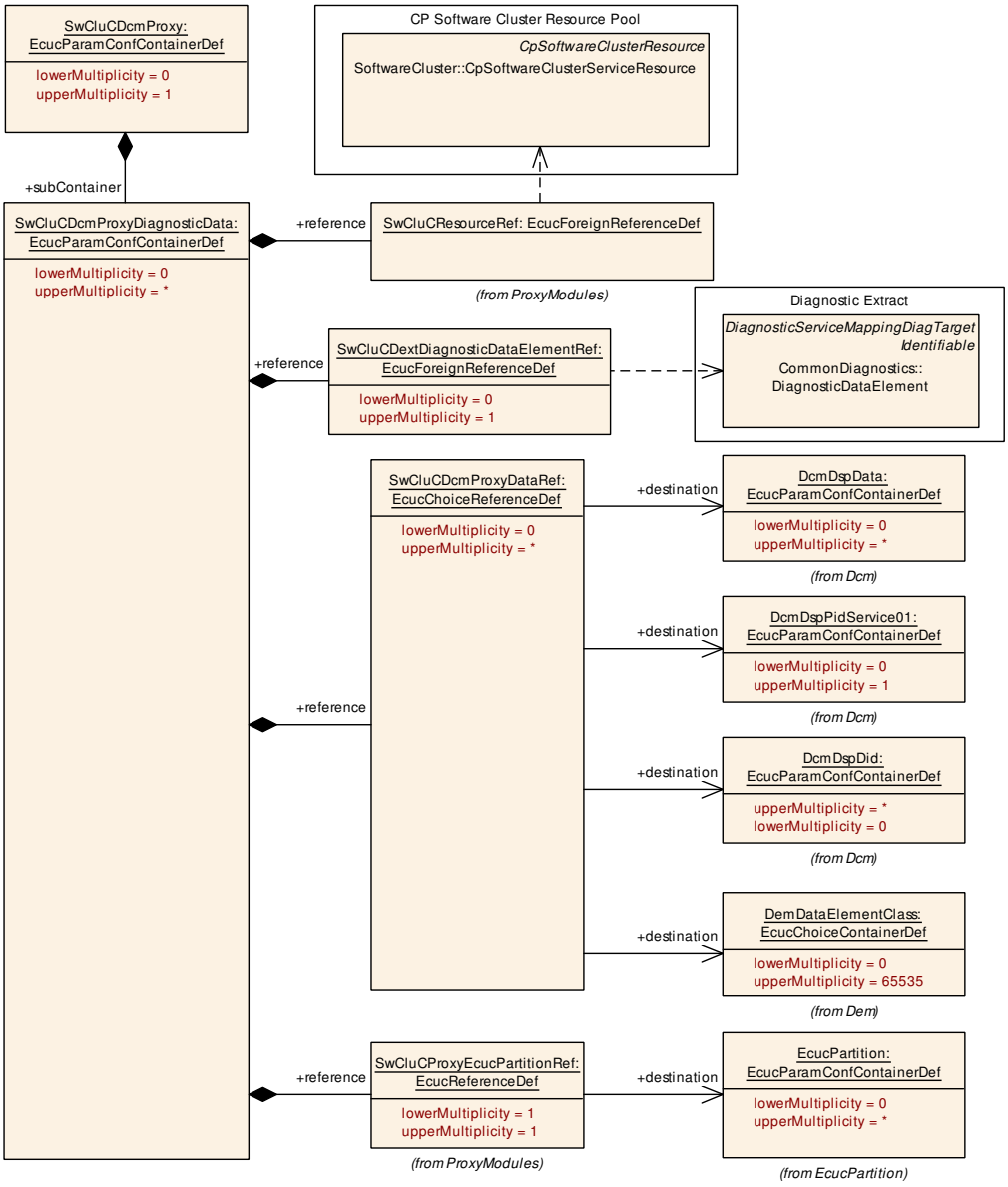

Figure 10.37: Dcm Proxy Diagnostic Data

[ECUC_SwCluC_00122] Definition of EcucParamConfContainerDef SwCluCDcm ProxyDiagnosticData ⌈

Container Name	SwCluCDcmProxyDiagnosticData
Parent Container	SwCluCDcmProxy
Upstream mapping defined	–
Description	Configures a Diagnostic Data proxy for a DID or PID or IOControl.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCDcmProxyDataRef	0..*	[ECUC_SwCluC_00123]
SwCluCDextDiagnosticDataElementRef	0..1	[ECUC_SwCluC_00138]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00123] Definition of EcucChoiceReferenceDef SwCluCDcmProxyDataRef ⌈

Parameter Name	SwCluCDcmProxyDataRef
Parent Container	SwCluCDcmProxyDiagnosticData
Upstream mapping defined	–
Description	Reference to the diagnostic data of Dcm which can either be a DcmDspData, or a Dcm DspPidService01, or a DcmDspDid. The DcmDspDid reference is only applicable in the Application Software Cluster if the DID has exactly one Diagnostic Data Element requiring the DID based interface. The reference is mandatory in the Application Software Cluster and can be used in the Host Software Cluster to identify the related diagnostic data in the Dcm configuration.
Multiplicity	0..*
Type	Choice reference to [DcmDspData, DcmDspDid, DcmDspPidService01, DemDataElementClass]
Pre-Compile only	true

⌋

[ECUC_SwCluC_00138] Definition of EcucForeignReferenceDef SwCluCDextDiagnosticDataElementRef ⌈

Parameter Name	SwCluCDextDiagnosticDataElementRef
Parent Container	SwCluCDcmProxyDiagnosticData
Upstream mapping defined	–
Description	Reference to the DiagnosticDataElement in DEXT for back-tracing in case of DEXT based configuration approaches
Multiplicity	0..1
Type	Foreign reference to DIAGNOSTIC-DATA-ELEMENT
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02427] `SwCluCDcmProxyDiagnosticData` **relates only to a** `CpSoftwareClusterServiceResource` **of** `category SWCLUSTER_RES_DCM_DIAGNOSTIC_DATA`

Upstream requirements: SRS_SwCluC_00208, SRS_BSW_00167

⌈The `SwCluCDcmProxyDiagnosticData.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DCM_DIAGNOSTIC_DATA`.⌋

10.2.7.4.3 Dcm Proxy Routine Control

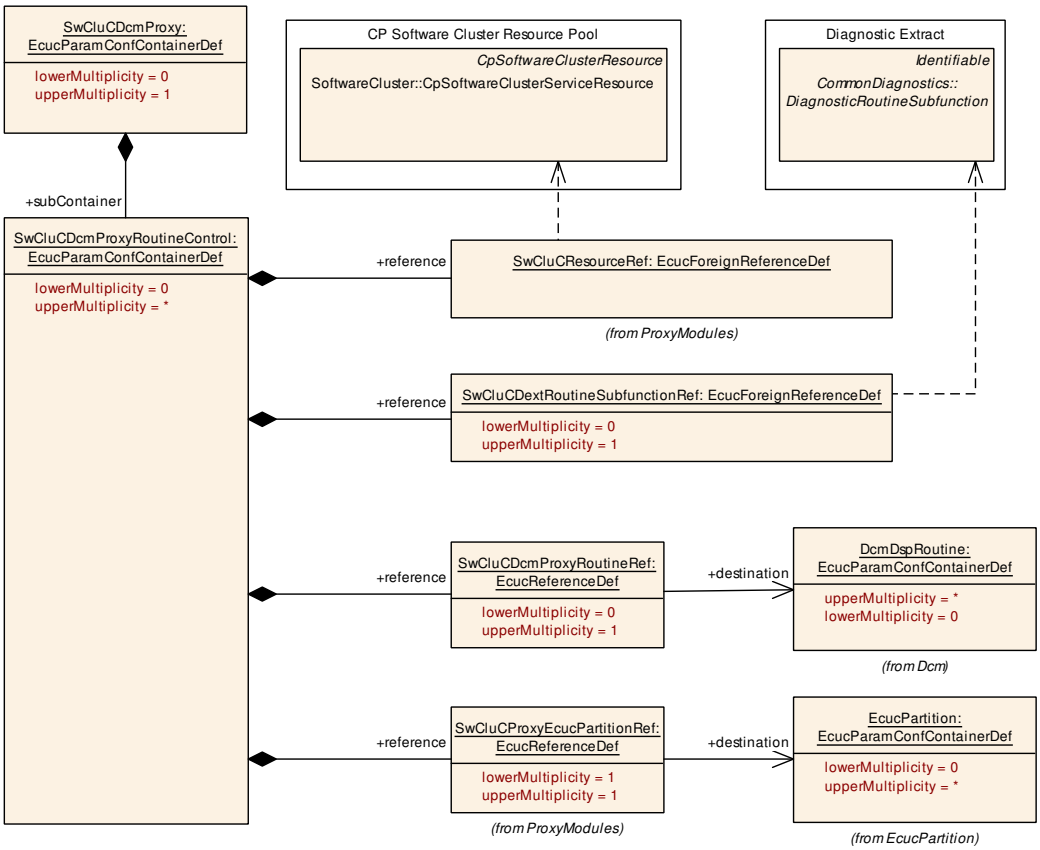

Figure 10.38: Dcm Proxy Routine Control

[ECUC_SwCluC_00133] Definition of EcucParamConfContainerDef SwCluCDcm ProxyRoutineControl ⌈

Container Name	SwCluCDcmProxyRoutineControl
Parent Container	SwCluCDcmProxy
Upstream mapping defined	–
Description	Configures a Routine Control proxy.
Multiplicity	0..*

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCDcmProxyRoutineRef	0..1	[ECUC_SwCluC_00136]
SwCluCDextRoutineSubfunctionRef	0..1	[ECUC_SwCluC_00135]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00136] Definition of EcucReferenceDef SwCluCDcmProxyRoutineRef ⌈

Parameter Name	SwCluCDcmProxyRoutineRef
Parent Container	SwCluCDcmProxyRoutineControl
Upstream mapping defined	–
Description	Reference to the routine configuration of Dcm. The reference is mandatory in the Application Software Cluster and can be used in the Host Software Cluster to identify the related routine control in the Dcm configuration.
Multiplicity	0..1
Type	Reference to DcmDspRoutine
Pre-Compile only	true

⌋

[ECUC_SwCluC_00135] Definition of EcucForeignReferenceDef SwCluCDextRoutineSubfunctionRef ⌈

Parameter Name	SwCluCDextRoutineSubfunctionRef
Parent Container	SwCluCDcmProxyRoutineControl
Upstream mapping defined	–
Description	Reference to the Routine Subfunction in DEXT for back-tracing in case of DEXT based configuration approaches
Multiplicity	0..1
Type	Foreign reference to DIAGNOSTIC-ROUTINE-SUBFUNCTION
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02428] **SwCluCDcmProxyRoutineControl relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_DCM_ROUTINE_CONTROL**

Upstream requirements: SRS_SwCluC_00208, SRS_BSW_00167

⌈The SwCluCDcmProxyRoutineControl.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_DCM_ROUTINE_CONTROL.⌋

10.2.7.4.4 Dcm Proxy Service Request Notification

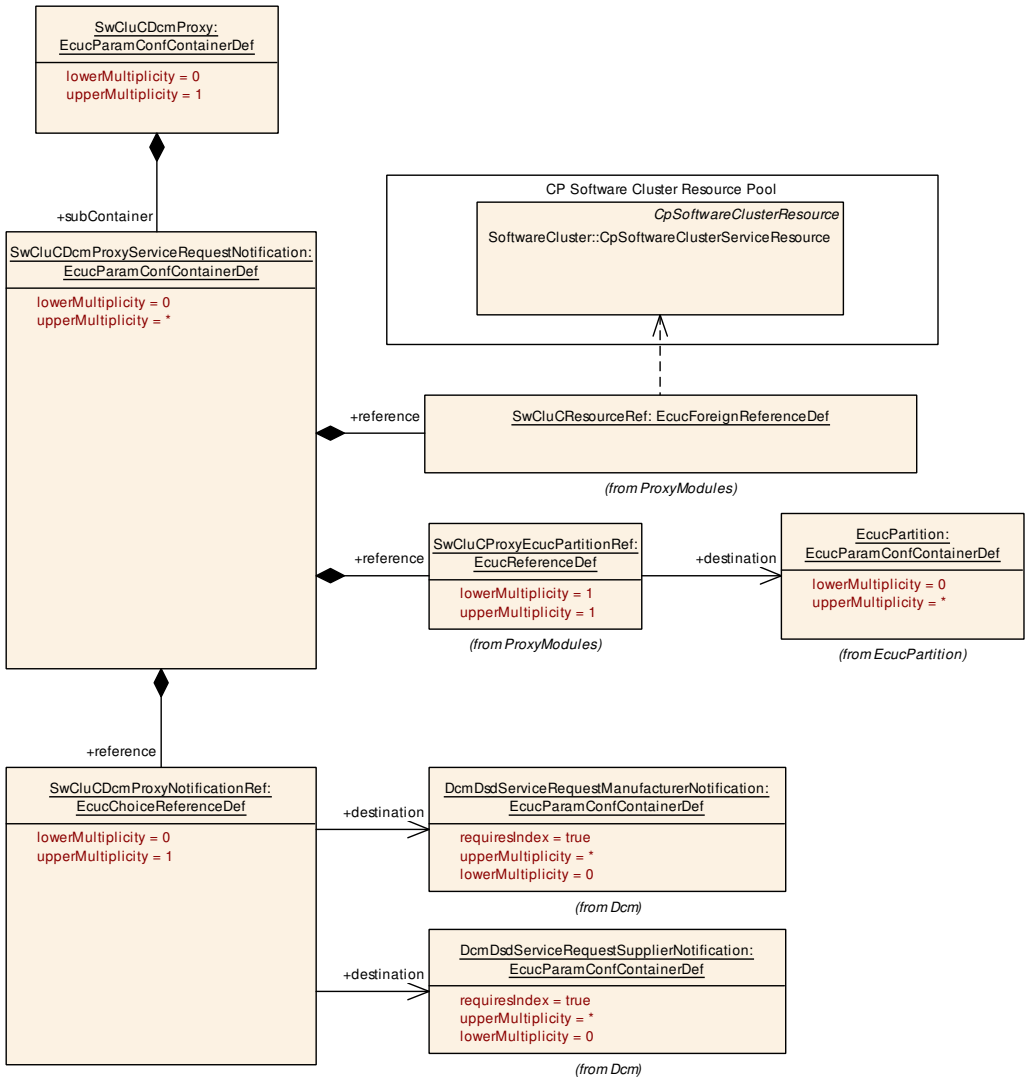

Figure 10.39: Dcm Proxy Service Request Notification

[ECUC_SwCluC_00145] Definition of EcucParamConfContainerDef SwCluCDcm ProxyServiceRequestNotification ⌈

Container Name	SwCluCDcmProxyServiceRequestNotification
Parent Container	SwCluCDcmProxy
Upstream mapping defined	–
Description	Configures a Service Request Notification proxy.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCDcmProxyNotificationRef	0..1	[ECUC_SwCluC_00146]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00146] Definition of EcucChoiceReferenceDef SwCluCDcmProxyNotificationRef ⌈

Parameter Name	SwCluCDcmProxyNotificationRef
Parent Container	SwCluCDcmProxyServiceRequestNotification
Upstream mapping defined	–
Description	Reference to the Service Request Notifcation configuration of Dcm which can either be a DcmDsdServiceRequestSupplierNotification or DcmDsdServiceRequest ManufacturerNotification The reference is mandatory in the Application Software Cluster and can be used in the Host Software Cluster to identify the related notification in the Dcm configuration.
Multiplicity	0..1
Type	Choice reference to [DcmDsdServiceRequestManufacturerNotification, DcmDsd ServiceRequestSupplierNotification]
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02429] `SwCluCDcmProxyServiceRequestNotification` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DCM_SERVICE_REQUEST_NOTIFICATION`

Upstream requirements: SRS_SwCluC_00208, SRS_BSW_00167

⌈The `SwCluCDcmProxyServiceRequestNotification.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DCM_SERVICE_REQUEST_NOTIFICATION`.⌋

10.2.7.5 Dem Proxy

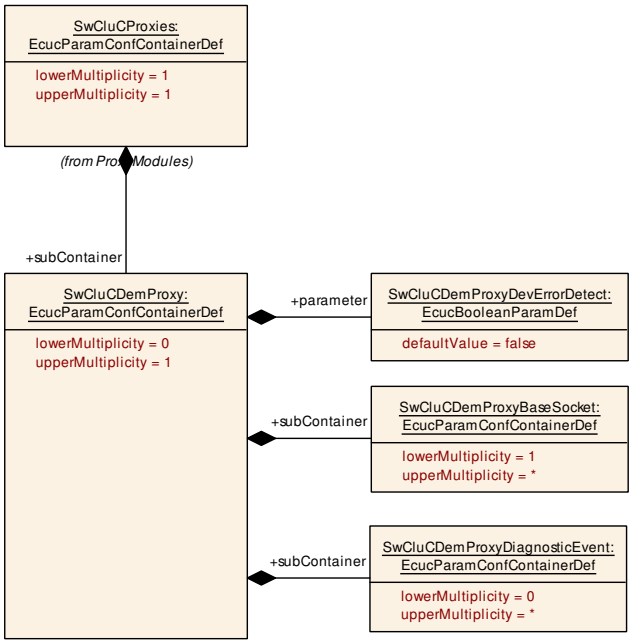

Figure 10.40: Dem Proxy Parameter

[ECUC_SwCluC_00112] Definition of EcucParamConfContainerDef SwCluCDem Proxy ⌈

Container Name	SwCluCDemProxy
Parent Container	SwCluCProxies
Upstream mapping defined	–
Description	DemProxy specific parameters.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCDemProxyDevErrorDetect	1	[ECUC_SwCluC_00113]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCDemProxyBaseSocket	1..*	This container configures how many EcucPartitions specific API links are required for the Dem Proxy and to which cluster resource the API set belongs. In the Dem Low Proxy all offered API sets needs to be configured. In the Dem High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the Ecuc Partition on which the API set is provided.
SwCluCDemProxyDiagnosticEvent	0..*	Configures a Diagnostic Event proxy.

⌋

[ECUC_SwCluC_00113] Definition of EcucBooleanParamDef SwCluCDemProxy DevErrorDetect ⌈

Parameter Name	SwCluCDemProxyDevErrorDetect
Parent Container	SwCluCDemProxy
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

⌋

10.2.7.5.1 Dem Base Socket

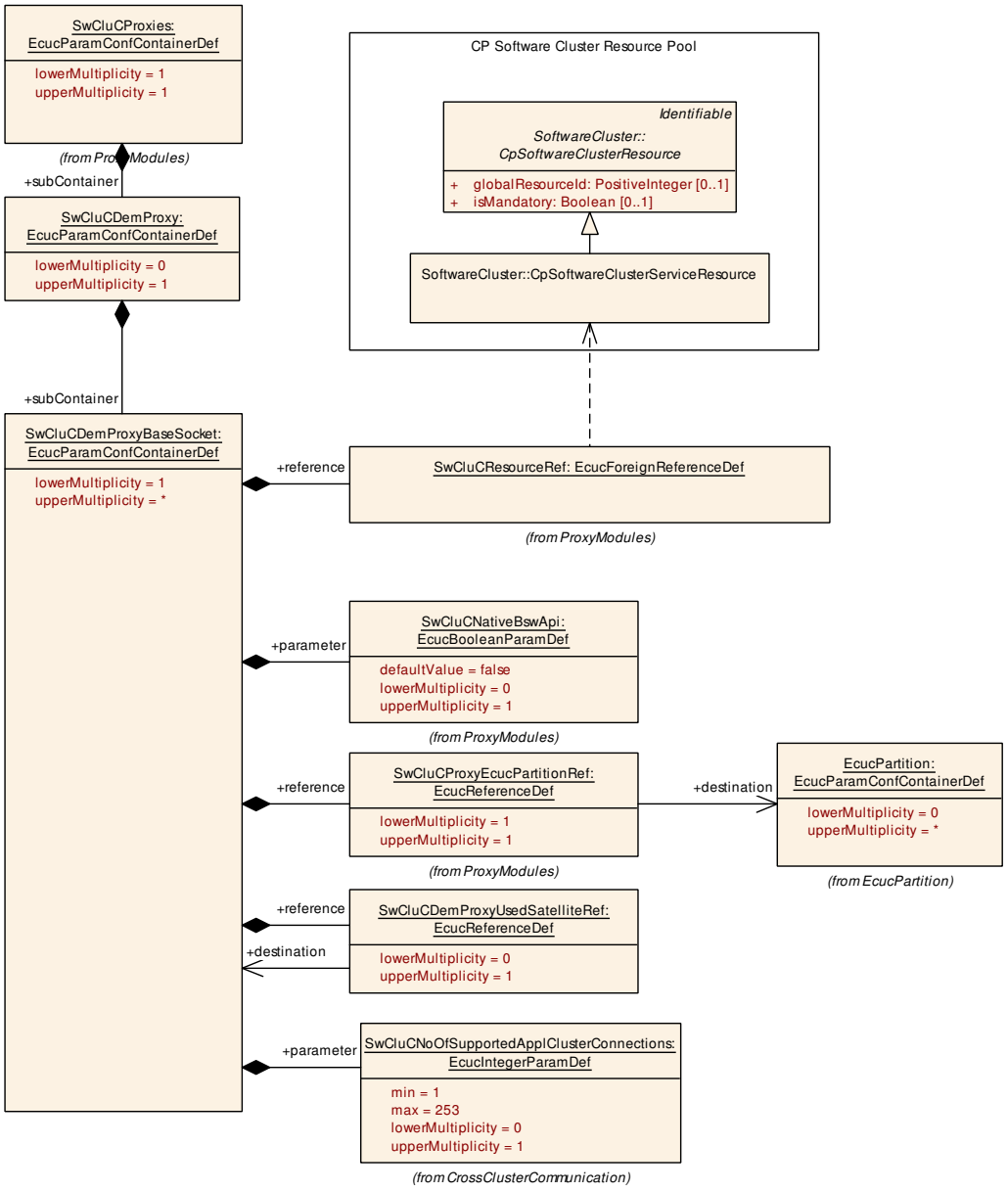

Figure 10.41: Dem Proxy Base Socket

[ECUC_SwCluC_00114] Definition of EcucParamConfContainerDef SwCluCDem ProxyBaseSocket ⌈

Container Name	SwCluCDemProxyBaseSocket
Parent Container	SwCluCDemProxy
Upstream mapping defined	–

▽

△

Description	This container configures how many EcucPartitions specific API links are required for the Dem Proxy and to which cluster resource the API set belongs. In the Dem Low Proxy all offered API sets needs to be configured. In the Dem High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the API set is provided.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNativeBswApi	0..1	[ECUC_SwCluC_00058]
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCDemProxyUsedSatelliteRef	0..1	[ECUC_SwCluC_00115]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00058] SwCluCNativeBswApi, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00166] SwCluCNoOfSupportedApplClusterConnections, see definition below container SwCluCComProxyBaseSocket.

[ECUC_SwCluC_00115] Definition of EcucReferenceDef SwCluCDemProxyUsedSatelliteRef ⌈

Parameter Name	SwCluCDemProxyUsedSatelliteRef
Parent Container	SwCluCDemProxyBaseSocket
Upstream mapping defined	–
Description	Reference to the SwCluCDemBaseSocket which has access to a satellite of the Dem. The owning SwCluCDemBaseSocket uses the Dem satellite of the referenced SwCluCDemBaseSocket.
Multiplicity	0..1
Type	Reference to SwCluCDemProxyBaseSocket
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02720] `SwCluCDemProxyBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DEM_BASE_SOCKET`

Upstream requirements: SRS_SwCluC_00209, SRS_BSW_00167

⌈The `SwCluCDemProxyBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_DEM_BASE_SOCKET`.⌋

10.2.7.5.2 Dem Proxy Diagnostic Event

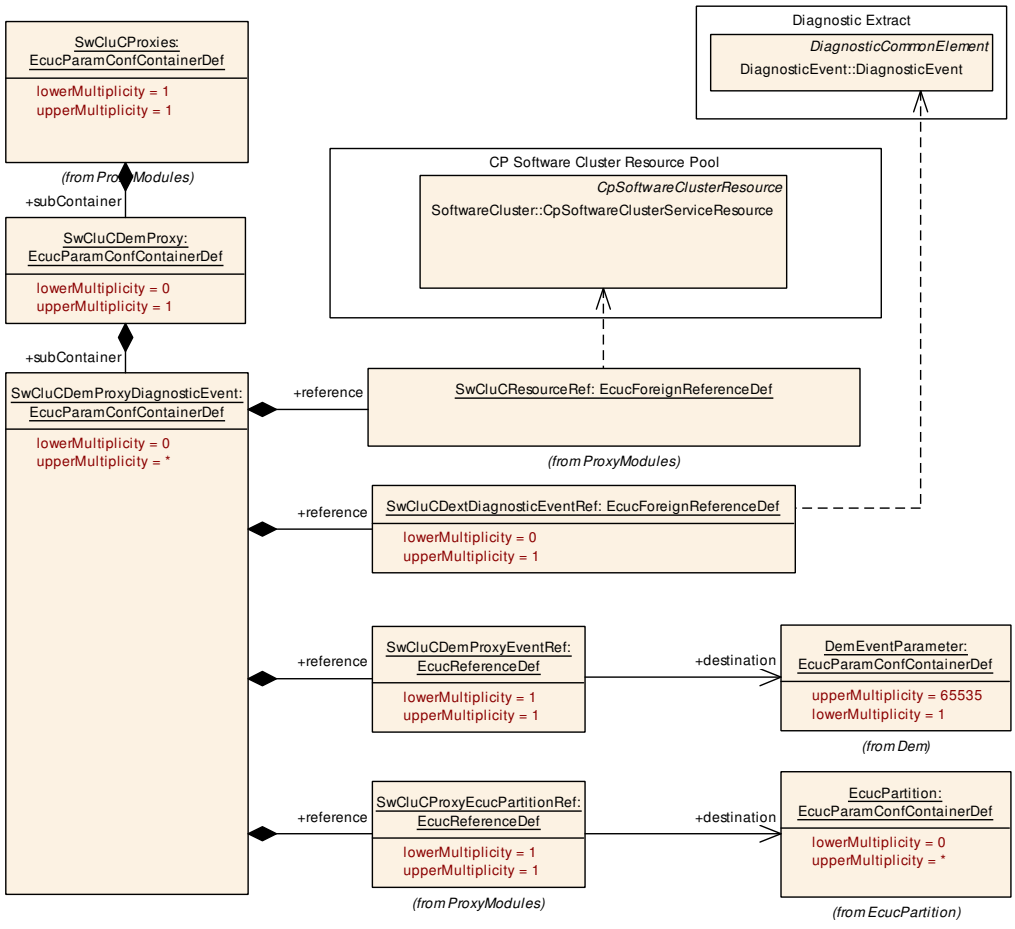

Figure 10.42: Dem Proxy Diagnostic Event

[ECUC_SwCluC_00116] Definition of EcucParamConfContainerDef SwCluCDem ProxyDiagnosticEvent ⌈

Container Name	SwCluCDemProxyDiagnosticEvent
Parent Container	SwCluCDemProxy
Upstream mapping defined	–

▽

△

Description	Configures a Diagnostic Event proxy.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCDemProxyEventRef	1	[ECUC_SwCluC_00117]
SwCluCDextDiagnosticEventRef	0..1	[ECUC_SwCluC_00140]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00117] Definition of EcucReferenceDef SwCluCDemProxyEventRef ⌈

Parameter Name	SwCluCDemProxyEventRef
Parent Container	SwCluCDemProxyDiagnosticEvent
Upstream mapping defined	–
Description	Reference to the diagnostic event of Dem
Multiplicity	1
Type	Reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_SwCluC_00140] Definition of EcucForeignReferenceDef SwCluCDextDiagnosticEventRef ⌈

Parameter Name	SwCluCDextDiagnosticEventRef
Parent Container	SwCluCDemProxyDiagnosticEvent
Upstream mapping defined	–
Description	Reference to the Diagnostic Event in DEXT for back-tracing in case of DEXT based configuration approaches
Multiplicity	0..1
Type	Foreign reference to DIAGNOSTIC-EVENT
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_02721] **SwCluCDemProxyDiagnosticEvent relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_DEM_DIAGNOSTIC_EVENT**

Upstream requirements: SRS_SwCluC_00209, SRS_BSW_00167

⌈The SwCluCDemProxyDiagnosticEvent.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_DEM_DIAGNOSTIC_EVENT.⌋

10.2.7.6 FiM Proxy

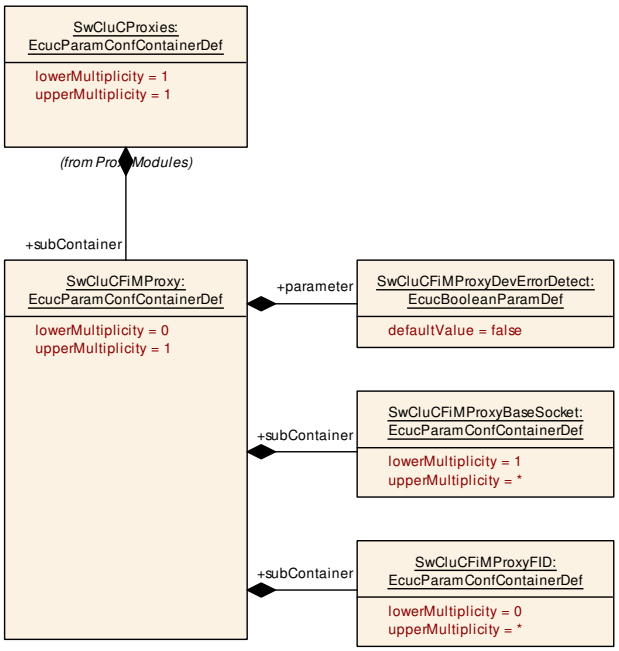

Figure 10.43: FiM Proxy Parameter

[ECUC_SwCluC_00105] Definition of EcucParamConfContainerDef SwCluCFiMProxy ⌈

Container Name	SwCluCFiMProxy
Parent Container	SwCluCProxies
Upstream mapping defined	–
Description	FiM Proxy specific parameters.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCFiMProxyDevErrorDetect	1	[ECUC_SwCluC_00106]

Included Containers		
Container Name	**Multiplicity**	**Description**
SwCluCFiMProxyBaseSocket	1..*	This container configures how many EcucPartitions specific API links are required for the FiM Proxy and to which cluster resource the API set belongs. In the FiM Low Proxy all offered API sets needs to be configured. In the FiM High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the Ecuc Partition on which the API set is provided.
SwCluCFiMProxyFID	0..*	Configures a Function Inhibition proxy.

⌋

[ECUC_SwCluC_00106] Definition of EcucBooleanParamDef SwCluCFiMProxy DevErrorDetect ⌈

Parameter Name	SwCluCFiMProxyDevErrorDetect
Parent Container	SwCluCFiMProxy
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

⌋

10.2.7.6.1 FiM Base Socket

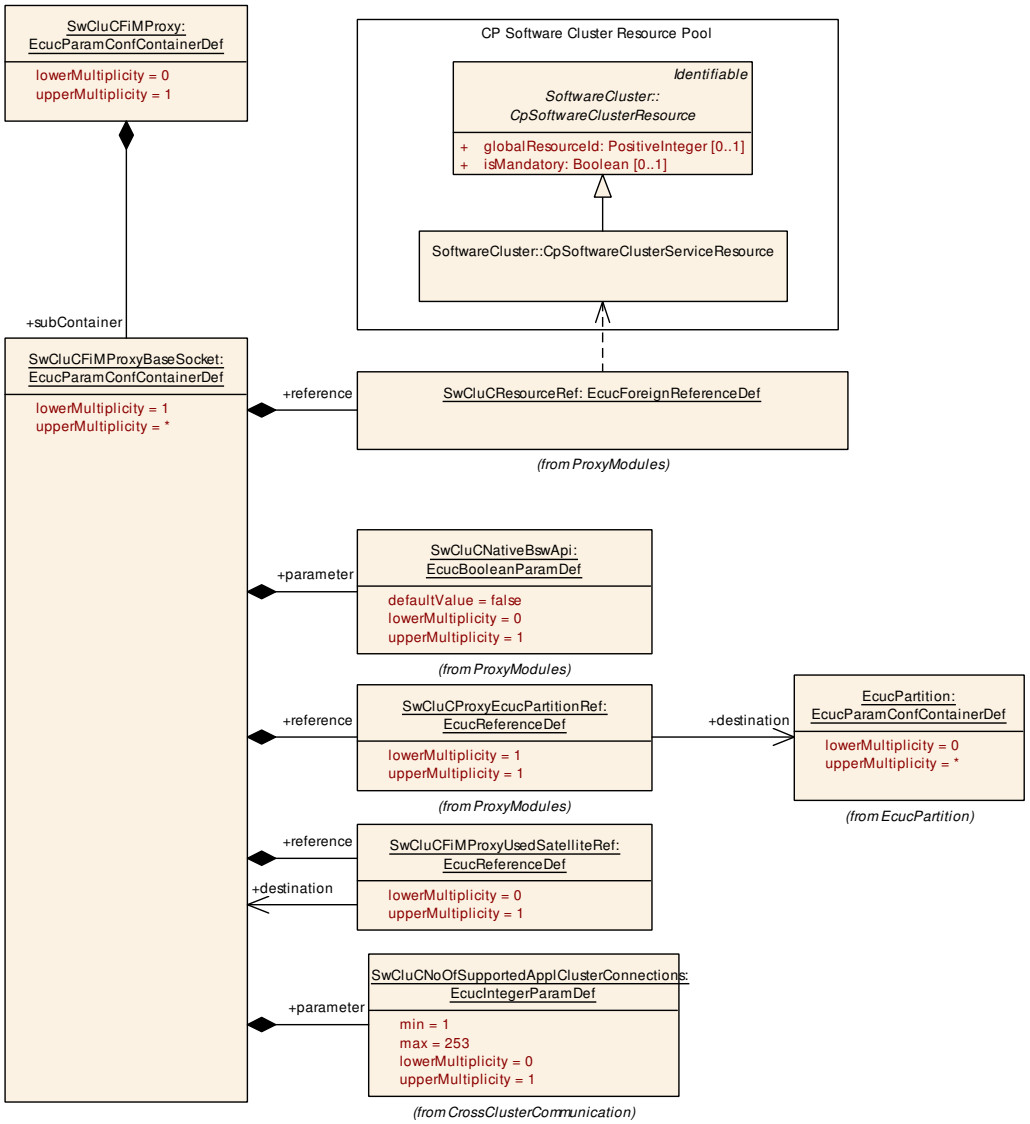

Figure 10.44: FiM Proxy Base Socket

[ECUC_SwCluC_00107] Definition of EcucParamConfContainerDef SwCluCFiMProxyBaseSocket ⌈

Container Name	SwCluCFiMProxyBaseSocket
Parent Container	SwCluCFiMProxy
Upstream mapping defined	–

▽

△

Description	This container configures how many EcucPartitions specific API links are required for the FiM Proxy and to which cluster resource the API set belongs. In the FiM Low Proxy all offered API sets needs to be configured. In the FiM High Proxy only the ones are configured which are offered inside this Application Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the API set is provided.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SwCluCNativeBswApi	0..1	[ECUC_SwCluC_00058]
SwCluCNoOfSupportedApplClusterConnections	0..1	[ECUC_SwCluC_00166]
SwCluCFiMProxyUsedSatelliteRef	0..1	[ECUC_SwCluC_00108]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

For parameter table [ECUC_SwCluC_00058] SwCluCNativeBswApi, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00166] SwCluCNoOfSupportedApplClusterConnections, see definition below container SwCluCComProxyBaseSocket.

[ECUC_SwCluC_00108] Definition of EcucReferenceDef SwCluCFiMProxyUsedSatelliteRef ⌈

Parameter Name	SwCluCFiMProxyUsedSatelliteRef
Parent Container	SwCluCFiMProxyBaseSocket
Upstream mapping defined	–
Description	Reference to the SwCluCFiMBaseSocket which has access to a satellite of the FiM. The owning SwCluCFiMBaseSocket uses the FiM satellite of the referenced SwCluCFiMBaseSocket.
Multiplicity	0..1
Type	Reference to SwCluCFiMProxyBaseSocket
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_03210] `SwCluCFiMProxyBaseSocket` relates only to a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_FIM_BASE_SOCKET`

Upstream requirements: SRS_SwCluC_00210, SRS_BSW_00167

⌈The `SwCluCFiMProxyBaseSocket.SwCluCResourceRef` shall only reference a `CpSoftwareClusterServiceResource` of `category SWCLUSTER_RES_FIM_BASE_SOCKET`.⌋

10.2.7.6.2 FiM Proxy FID

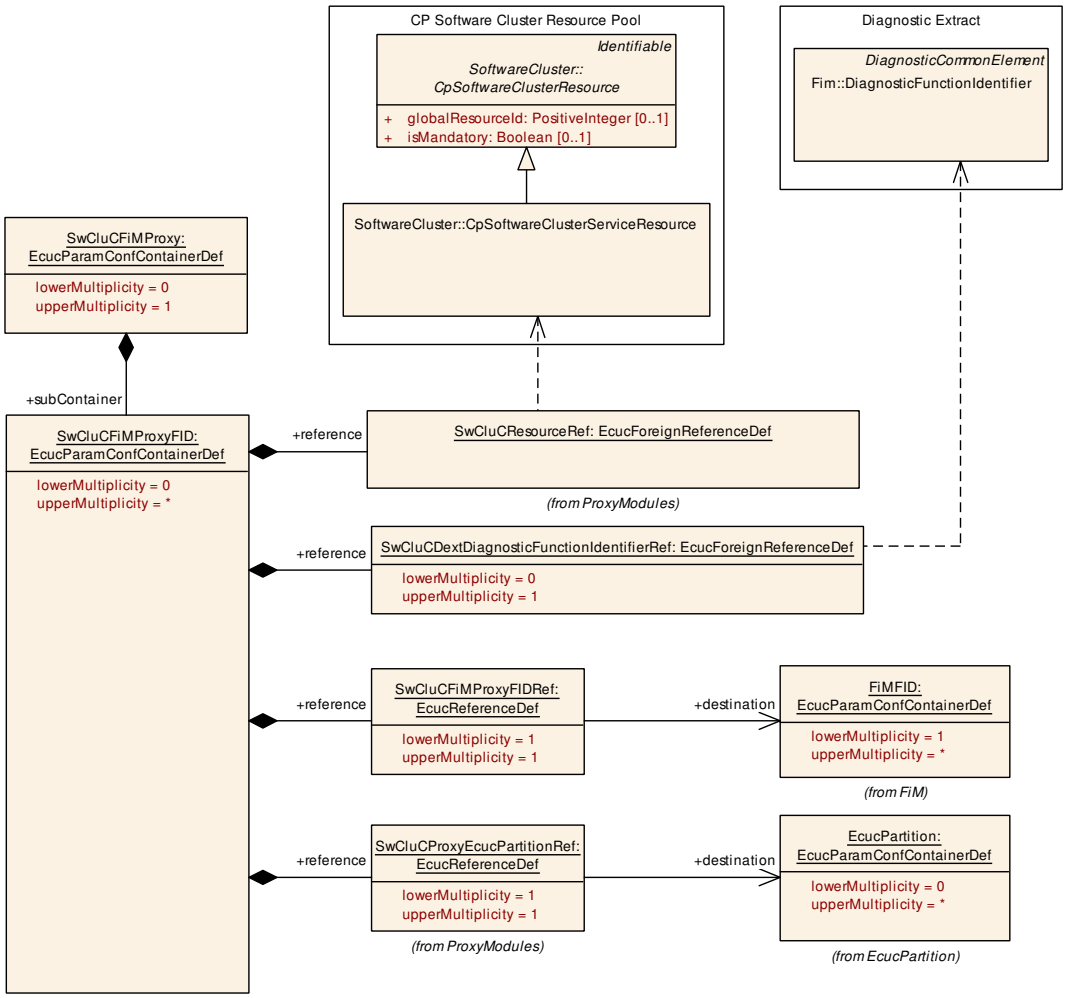

Figure 10.45: FiM Proxy FID

[ECUC_SwCluC_00109] Definition of EcucParamConfContainerDef SwCluCFiMProxyFID ⌈

Container Name	SwCluCFiMProxyFID
Parent Container	SwCluCFiMProxy
Upstream mapping defined	–
Description	Configures a Function Inhibition proxy.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SwCluCDextDiagnosticFunctionIdentifierRef	0..1	[ECUC_SwCluC_00142]
SwCluCFiMProxyFIDRef	1	[ECUC_SwCluC_00110]
SwCluCProxyEcucPartitionRef	1	[ECUC_SwCluC_00059]
SwCluCResourceRef	1	[ECUC_SwCluC_00087]

No Included Containers

⌋

[ECUC_SwCluC_00142] Definition of EcucForeignReferenceDef SwCluCDextDiagnosticFunctionIdentifierRef ⌈

Parameter Name	SwCluCDextDiagnosticFunctionIdentifierRef
Parent Container	SwCluCFiMProxyFID
Upstream mapping defined	–
Description	Reference to the DiagnosticFunctionIdentifier in DEXT for back-tracing in case of DEXT based configuration approaches
Multiplicity	0..1
Type	Foreign reference to DIAGNOSTIC-FUNCTION-IDENTIFIER
Pre-Compile only	true

⌋

[ECUC_SwCluC_00110] Definition of EcucReferenceDef SwCluCFiMProxyFIDRef ⌈

Parameter Name	SwCluCFiMProxyFIDRef
Parent Container	SwCluCFiMProxyFID
Upstream mapping defined	–
Description	Reference to the FIMFID
Multiplicity	1
Type	Reference to FiMFID
Pre-Compile only	true

⌋

For parameter table [ECUC_SwCluC_00059] SwCluCProxyEcucPartitionRef, see definition below container SwCluCDcmProxyBaseSocket.

For parameter table [ECUC_SwCluC_00087] SwCluCResourceRef, see definition below container SwCluCBaseConfigurationCheck.

[SWS_SwCluC_CONSTR_03211] **SwCluCFiMProxyFID relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_FIM_FID**

Upstream requirements: SRS_SwCluC_00210, SRS_BSW_00167

⌈The SwCluCFiMProxyFID.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_FIM_FID.⌋

10.3 Published Information

For details refer to [3] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[SWS_SwCluC_NA_00001] Requirements for software components Not Applicable to this Specification

Upstream requirements: SRS_BSW_00168, SRS_BSW_00170

⌈SwCluC is not a software component.⌋

[SWS_SwCluC_NA_00002] Requirement on wake-up reason reporting Not Applicable to this Specification

Upstream requirements: SRS_BSW_00375

⌈SwCluC does not support wakeup.⌋

[SWS_SwCluC_NA_00003] General requirements on Basic Software Module configuration Not Applicable to this Specification

Upstream requirements: SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00419

⌈General ECU-C principal applied on specified configuration definition.⌋

[SWS_SwCluC_NA_00004] Requirement on error detection Not Applicable to this Specification

Upstream requirements: SRS_BSW_00386

⌈SwCluC does not specify conditions on error detections.⌋

[SWS_SwCluC_NA_00005] Requirement on configurable initialization sequence Not Applicable to this Specification

Upstream requirements: SRS_BSW_00416

⌈Initialization sequence is controlled by BswM and SwCluC does not specify additional configuration parameters.⌋

[SWS_SwCluC_NA_00006] Requirement on error event reporting Not Applicable to this Specification

Upstream requirements: SRS_BSW_00417, SRS_BSW_00422

⌈SwCluC does not report diagnostic events on its own.⌋

[SWS_SwCluC_NA_00007] Requirement on BSW module description template Not Applicable to this Specification

Upstream requirements: SRS_BSW_00425

⌈Requirement on BSWMD-T are not handled in this specification.⌋

[SWS_SwCluC_NA_00008] Requirement on separate main processing functions Not Applicable to this Specification

Upstream requirements: SRS_BSW_00432

⌈SwCluC does not specify general data transmission APIs.⌋

[SWS_SwCluC_NA_00009] Requirement on generic module interfaces Not Applicable to this Specification

Upstream requirements: SRS_BSW_00461

⌈SwCluC does not specify generic module interfaces.⌋

[SWS_SwCluC_NA_00010] Requirement on production errors and extended production errors reporting Not Applicable to this Specification

Upstream requirements: SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472

⌈SwCluC does not specify neither production errors nor extended production errors.⌋

[SWS_SwCluC_NA_00011] Requirement on security event reporting Not Applicable to this Specification

Upstream requirements: SRS_BSW_00490, SRS_BSW_00491

⌈SwCluC does not specify no security events.⌋

[SWS_SwCluC_NA_00012] Requirement on Implementation source files Not Applicable to this Specification

Upstream requirements: SRS_BSW_00004

⌈Only header files are defined for SwCluC; no dedicated source-code files are specified.⌋

B Referenced Meta Classes

Class	***AbstractAccessPoint*** (abstract)			
Note	Abstract class indicating an access point from an `ExecutableEntity`.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AsynchronousServerCallResultPoint, ExternalTriggeringPointIdent, InternalTriggeringPoint, ModeAccessPointIdent, ModeSwitchPoint, ParameterAccess, *ServerCallPoint*, VariableAccess			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
returnValue Provision	RteApiReturnValue ProvisionEnum	0..1	attr	This attribute controls the provision of return values for RTE APIs that correspond to the enclosing access point. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.1: AbstractAccessPoint

Class	**ApplicationArrayDataType**			
Note	An application data type which is an array, each element is of the same application data type. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *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 if it is a variable size array.
element	ApplicationArray Element	0..1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine. **Tags:** atp.Splitkey=element, element.shortName

Table B.2: ApplicationArrayDataType

Class	**ApplicationArrayElement**			
Note	Describes the properties of the elements of an application array data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationArrayDataType.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	ApplicationPrimitive DataType	0..1	ref	This reference can be taken to assign a `CompuMethod` of category `TEXTTABLE` to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name. **Tags:** atp.Splitkey=indexDataType



△

Class	ApplicationArrayElement			
maxNumberOf Elements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table B.3: ApplicationArrayElement

Class	*ApplicationCompositeElementDataPrototype* (abstract)			
Note	This class represents a data prototype which is aggregated within a composite application data type (record or array). It is introduced to provide a better distinction between target and context in instance Refs.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApplicationArrayElement, ApplicationRecordElement			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	ApplicationDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table B.4: ApplicationCompositeElementDataPrototype

Class	*ApplicationDataType* (abstract)			
Note	`ApplicationDataType` defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An `ApplicationDataType` represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using `ApplicationDataType`s only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.5: ApplicationDataType

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 B.6: ApplicationError

Class	ApplicationPrimitiveDataType			
Note	A primitive data type defines a set of allowed values. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.7: ApplicationPrimitiveDataType

Class	ApplicationRecordDataType			
Note	An application data type which can be decomposed into prototypes of other application data types. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *Atp Classifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element (ordered)	ApplicationRecord Element	*	aggr	Specifies an element of a record. The aggregation of `ApplicationRecordElement` is subject to variability with the purpose to support the conditional existence of elements inside a `ApplicationRecordData` Type. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table B.8: ApplicationRecordDataType

Class	ApplicationRecordElement			
Note	Describes the properties of one particular element of an application record data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationRecordDataType.element, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing `ApplicationRecordElement` as optional. This means the that, at runtime, the `ApplicationRecordElement` may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the `ApplicationRecordElement` as not valid at the sending end of a communication and determine its validity at the receiving end.

Table B.9: ApplicationRecordElement

Class	ArgumentDataPrototype
Note	An argument of an operation, carries direction and implementation information.
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*
Aggregated by	*AtpClassifier*.atpFeature, ClientServerOperation.argument

▽

△

Class	**ArgumentDataPrototype**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirection Enum	0..1	attr	This attribute specifies the direction of the argument.
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	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 B.10: ArgumentDataPrototype

Enumeration	**ArraySizeSemanticsEnum**
Note	This enumeration provides options for how the number of elements in an array object shall be interpreted.
Aggregated by	ApplicationArrayElement.arraySizeSemantics, DiagnosticDataElement.arraySizeSemantics, ImplementationDataTypeElement.arraySizeSemantics, SwTextProps.arraySizeSemantics
Literal	***Description***
fixedSize	This means that the array object will always have a fixed number of elements. **Tags:** atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the array object might vary at run-time. The configured size represents the maximum number of elements in the array object. **Tags:** atp.EnumerationLiteralIndex=1

Table B.11: ArraySizeSemanticsEnum

Class	**AssemblySwConnector**			
Note	`AssemblySwConnector`s are exclusively used to connect `SwComponentPrototype`s in the context of a `CompositionSwComponentType`.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **Tags:** atp.Splitkey=provider, provider.contextComponent, provider.targetPPort **InstanceRef implemented by:** PPortInComposition InstanceRef
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **Tags:** atp.Splitkey=requester, requester.context Component, requester.targetRPort **InstanceRef implemented by:** RPortInComposition InstanceRef

Table B.12: AssemblySwConnector

Class	**AsynchronousServerCallReturnsEvent**
Note	This event is raised when an asynchronous server call is finished.
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event

▽

△

Class	AsynchronousServerCallReturnsEvent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventSource	AsynchronousServer CallResultPoint	0..1	ref	The referenced AsynchronousServerCallResultPoint raises this AsynchronousServerCallReturnsEvent when the asynchronous server call returns. **Tags:** atp.Splitkey=eventSource This Attribute is only used by the AUTOSAR Classic Platform.

Table B.13: AsynchronousServerCallReturnsEvent

Class	*AutosarDataPrototype* (abstract)			
Note	Base class for prototypical roles of an `AutosarDataType`.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, ParameterDataPrototype, VariableDataPrototype			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	AutosarDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table B.14: AutosarDataPrototype

Class	*AutosarDataType* (abstract)			
Note	Abstract base class for user defined AUTOSAR data types for software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType. **Tags:** atp.Splitkey=swDataDefProps

Table B.15: AutosarDataType

Class	BackgroundEvent			
Note	This event is used to start `RunnableEntity`s that are supposed to be executed in the background.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.16: BackgroundEvent

Class	**BaseType** (abstract)			
Note	This abstract meta-class represents the ability to specify a platform dependent base type.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	SwBaseType			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table B.17: BaseType

Class	**BaseTypeDirectDefinition**			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Aggregated by	*BaseType*.baseTypeDefinition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
baseType Encoding	BaseTypeEncoding String	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. ▽

▽

△

Class	BaseTypeDirectDefinition			
				△ This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:** xml.sequenceOffset=120

Table B.18: BaseTypeDirectDefinition

Class	***BswEvent*** (abstract)			
Note	Base class of various kinds of events which are used to trigger a BswModuleEntity of this BSW module or cluster. The event is local to the BSW module or cluster. The short name of the meta-class instance is intended as an input to configure the required API of the BSW Scheduler.			
Base	*ARObject*, *AbstractEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswInterruptEvent, BswOperationInvokedEvent, *BswScheduleEvent*			
Aggregated by	BswInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Limitation	BswDistinguished Partition	*	ref	The existence of this reference indicates that the usage of the event is limited to the context of the referred Bsw DistinguishedPartitions. **Tags:** atp.Splitkey=contextLimitation
disabledInMode	ModeDeclaration	*	iref	The modes, in which this event is disabled. **Tags:** atp.Splitkey=disabledInMode, disabledIn Mode.contextModeDeclarationGroup, disabledIn Mode.targetMode **InstanceRef implemented by:** ModeInBswModule DescriptionInstanceRef
startsOnEvent	BswModuleEntity	0..1	ref	The entity which is started by the event. **Tags:** atp.Splitkey=startsOnEvent

Table B.19: BswEvent

Class	**BswExternalTriggerOccurredEvent**			
Note	A BswEvent resulting from a trigger released by another module or cluster.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
trigger	Trigger	0..1	ref	The trigger associated with this event. The trigger is external to this module. **Tags:** atp.Splitkey=trigger

Table B.20: BswExternalTriggerOccurredEvent

Class	**BswInternalTriggerOccurredEvent**
Note	A BswEvent, which can happen sporadically. The event is activated by explicit calls from the module to the BSW Scheduler. The main purpose for such an event is to cause a context switch, e.g. from an ISR context into a task context. Activation and switching are handled within the same module or cluster only.
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Aggregated by	BswInternalBehavior.event

▽

△

Class	BswInternalTriggerOccurredEvent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventSource	BswInternalTriggering Point	0..1	ref	The activation point is the source of this event. **Tags:** atp.Splitkey=eventSource This Attribute is only used by the AUTOSAR Classic Platform.

Table B.21: BswInternalTriggerOccurredEvent

Class	BswModeSwitchEvent			
Note	A BswEvent resulting from a mode switch.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activation	ModeActivationKind	0..1	attr	Kind of activation w.r.t. to the referred mode.
mode (ordered)	ModeDeclaration	0..2	iref	Reference to one or two Modes that initiate the Mode Switch Event. **InstanceRef implemented by:** ModeInBswModule DescriptionInstanceRef

Table B.22: BswModeSwitchEvent

Class	BswModuleDescription			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:** atp.recommendedPackage=BswModuleDescriptions This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=bswModuleDependency, bswModule Dependency.shortName, bswModule Dependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, bswModule Documentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6

▽

△

Class	**BswModuleDescription**			
expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=expectedEntry, expectedEntry.bswModule Entry, expectedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=implementedEntry, implementedEntry.bsw ModuleEntry, implementedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is <<atp Splitable>>. **Tags:** atp.Splitkey=internalBehavior, internalBehavior.short Name xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:** xml.sequenceOffset=5
providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another partition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=providedClientServerEntry, providedClient ServerEntry.shortName, providedClientServer Entry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=providedData, providedData.shortName, providedData.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55

▽

△

Class	**BswModuleDescription**			
providedMode Group	ModeDeclarationGroup Prototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=providedModeGroup, providedMode Group.shortName, providedModeGroup.variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=releasedTrigger, releasedTrigger.shortName, releasedTrigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another partition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=requiredClientServerEntry, requiredClient ServerEntry.shortName, requiredClientServer Entry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The required Data is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=requiredData, requiredData.shortName, requiredData.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60

▽

△

Class	BswModuleDescription			
requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=requiredModeGroup, requiredMode Group.shortName, requiredModeGroup.variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=requiredTrigger, requiredTrigger.shortName, requiredTrigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table B.23: BswModuleDescription

Class	*BswModuleEntity* (abstract)			
Note	Specifies the smallest code fragment which can be described for a BSW module or cluster within AUTOSAR.			
Base	*ARObject*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswCalledEntity, BswInterruptEntity, BswSchedulableEntity			
Aggregated by	BswInternalBehavior.entity			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
accessedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is accessed via API call by this entity. It shall be a ModeDeclarationGroupPrototype required by this module or cluster. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=accessedModeGroup, accessedMode Group.modeDeclarationGroupPrototype, accessedMode Group.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
activationPoint	BswInternalTriggering Point	*	ref	Activation point used by the module entity to activate one or more internal triggers. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=activationPoint, activationPoint.bswInternal TriggeringPoint, activationPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	***BswModuleEntity*** (abstract)			
callPoint	BswModuleCallPoint	*	aggr	A call point used in the code of this entity. The variability of this association is especially targeted at debug scenarios: It is possible to have one variant calling into the AUTOSAR debug module and another one which doesn't. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=callPoint, callPoint.shortName, call Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive Point	BswVariableAccess	*	aggr	The data is received via the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataReceivePoint, dataReceivePoint.short Name, dataReceivePoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	BswVariableAccess	*	aggr	The data is sent via the BSW Scheduler. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataSendPoint, dataSendPoint.shortName, dataSendPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	0..1	ref	The entry which is implemented by this module entity. **Tags:** atp.Splitkey=implementedEntry This Attribute is only used by the AUTOSAR Classic Platform.
issuedTrigger	Trigger	*	ref	A trigger issued by this entity via BSW Scheduler API call. It shall be a BswTrigger released (i.e. owned) by this module or cluster. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=issuedTrigger, issuedTrigger.trigger, issued Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
managedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is managed by this entity. It shall be a ModeDeclarationGroupPrototype provided by this module or cluster. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=managedModeGroup, managedMode Group.modeDeclarationGroupPrototype, managedMode Group.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	0..1	ref	A prefix to be used in generated names for the Bsw ModuleScheduler in the context of this BswModuleEntity, for example entry point prototypes, macros for dealing with exclusive areas, header file names. Details are defined in the SWS RTE. The prefix supersedes default rules for the prefix of those names. **Tags:** atp.Splitkey=schedulerNamePrefix

Table B.24: BswModuleEntity

Class	BswSchedulableEntity			
Note	BSW module entity, which is designed for control by the BSW Scheduler. It may for example implement a so-called "main" function.			
Base	*ARObject*, *BswModuleEntity*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswInternalBehavior.entity			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.25: BswSchedulableEntity

Class	BswTimingEvent			
Note	A recurring BswEvent driven by a time period.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	BswInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
period	TimeValue	0..1	attr	Requirement for the time period (in seconds) by which this event is triggered.

Table B.26: BswTimingEvent

Class	BswVariableAccess			
Note	The presence of a BswVariableAccess implies that a BswModuleEntity needs access to a VariableData Prototype via the BSW Scheduler. The kind of access is specified by the role in which the class is used.			
Base	*ARObject*, *Referrable*			
Aggregated by	*BswModuleEntity*.dataReceivePoint, *BswModuleEntity*.dataSendPoint			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessed Variable	VariableDataPrototype	0..1	ref	The data accessed via the BSW Scheduler. **Tags:** atp.Splitkey=accessedVariable
context Limitation	BswDistinguished Partition	*	ref	The existence of this reference indicates that the variable is received resp. sent only in the context of the referred BswDistinguishedPartitions. **Tags:** atp.Splitkey=contextLimitation

Table B.27: BswVariableAccess

Class	ClientServerApplicationErrorMapping			
Note	This meta-class represents the ability to map `ApplicationError`s onto each other.			
Base	*ARObject*			
Aggregated by	ClientServerInterfaceMapping.errorMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
firstApplication Error	ApplicationError	0..1	ref	This represents the first ApplicationError in the context of the ClientServerApplicationErrorMapping. **Tags:** atp.Splitkey=firstApplicationError
second ApplicationError	ApplicationError	0..1	ref	This represents the second ApplicationError in the context of the ClientServerApplicationErrorMapping. **Tags:** atp.Splitkey=secondApplicationError

Table B.28: ClientServerApplicationErrorMapping

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.
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface. **Tags:** atp.Splitkey=possibleError, possibleError.short Name

Table B.29: 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 B.30: ClientServerOperation

Class	ClientServerOperationComProps			
Note	Defines additional attributes for the implementation of Client Server communication between software clusters This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *CpSoftwareClusterCommunicationResourceProps*			
Aggregated by	CpSoftwareClusterCommunicationResource.communicationResourceProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
queueLength	PositiveInteger	0..1	attr	Length of call request queue on the server side. The queue is implemented by the SwCluC. The value shall be greater or equal to 1. Setting the value of queueLength to 1 implies that incoming requests are rejected while another request that arrived earlier is being processed.

Table B.31: ClientServerOperationComProps

Class	ClientServerOperationMapping			
Note	Defines the mapping of two particular `ClientServerOperation`s in context of two different `ClientServerInterface`s.			
Base	*ARObject*			
Aggregated by	ClientServerInterfaceMapping.operationMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
argument Mapping	DataPrototypeMapping	*	aggr	Defines the mapping of two particular ArgumentData Prototypes with unequal names or unequal semantic (resolution or range) in context of Operations. **Tags:** atp.Splitkey=argumentMapping
firstOperation	ClientServerOperation	0..1	ref	First to-be-mapped ClientServerOperation of a Client ServerInterface. **Tags:** atp.Splitkey=firstOperation
firstToSecond Data Transformation	DataTransformation	0..1	ref	This reference indicates that a DataTransformation is intended in the context of the ClientServerOperation Mapping. **Tags:** atp.Splitkey=firstToSecondDataTransformation
second Operation	ClientServerOperation	0..1	ref	Second to-be-mapped ClientServerOperation of a Client ServerInterface. **Tags:** atp.Splitkey=secondOperation

Table B.32: ClientServerOperationMapping

Class	CompositionSwComponentType			
Note	A `CompositionSwComponentType` aggregates `SwComponentPrototype`s (that in turn are typed by `SwComponentType`)s as well as `SwConnector`s for primarily connecting `SwComponentPrototype`s among each others and towards the surface of the `CompositionSwComponentType`. By this means, a hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**CompositionSwComponentType**			
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of SwComponentPrototypes is resolved post-build, the deselected SwComponentPrototypes are still contained in the ECUs build but the instances are inactive in that they are not scheduled by the RTE. The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an empty CompositionSwComponentType to e.g. a supplier for filling the internal structure. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=component, component.shortName, component.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec. **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMappingSet to be applied for the used ApplicationDataTypes in PortInterfaces. Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementors mainly because the RTE generator is not concerned about the CompositionSwComponentTypes. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. **Tags:** atp.Splitkey=dataTypeMapping

▽

△

Class	**CompositionSwComponentType**			
instantiation RTEEventProps	InstantiationRTEEvent Props	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=instantiationRTEEventProps, instantiation RTEEventProps.shortLabel, instantiationRTEEvent Props.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime This Attribute is only used by the AUTOSAR Classic Platform.
physical Dimension Mapping	PhysicalDimension MappingSet	0..1	ref	This reference identifies the `PhysicalDimensionMappingSet` that is applicable in the context of the enclosing `CompositionSwComponentType`. The `PhysicalDimensionMapping`s contained in the `PhysicalDimensionMappingSet` shall be taken into account for the assessment of the compatibility of `PhysicalDimension`s in the context of creation of a `PortInterfaceMapping` in the scope of the `CompositionSwComponentType`. **Tags:** atp.Splitkey=physicalDimensionMapping

Table B.33: CompositionSwComponentType

Class	**Compu**			
Note	This meta-class represents the ability to express one particular computation.			
Base	*ARObject*			
Aggregated by	CompuMethod.compuInternalToPhys, CompuMethod.compuPhysToInternal			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuContent	CompuContent	0..1	aggr	This specifies the details of the computation. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
compuDefault Value	CompuConst	0..1	aggr	This property can be used to specify an output value for a conversion formula, if the value to be converted lies outside the plausibility limit. Although this is possible for all conversion formulae, it is especially valid for variables with tabular conversion formulae. **Tags:** atp.Splitkey=compuDefaultValue xml.sequenceOffset=70

Table B.34: Compu

Class	**CompuConstFormulaContent**			
Note	This meta-class represents the fact that the constant value of the computation method is represented by a variation point. This difference is due to compatibility with ASAM HDO.			
Base	*ARObject*, *CompuConstContent*			
Aggregated by	CompuConst.compuConstContentType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	CompuConstFormulaContent			
vf	Numerical	0..1	attr	Value calculated via a system constant. This element is included in every case where parameters should be generated from numerical values during compile time (not runtime!). Thus for example, the influence of the cylinder number on conversion formulae can be introduced in a repeatable manner. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=30

Table B.35: CompuConstFormulaContent

Class	CompuConstTextContent			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Aggregated by	CompuConst.compuConstContentType			
Attribute	Type	Mult.	Kind	Note
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

Table B.36: CompuConstTextContent

Class	CompuMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
compuInternal ToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** atp.Splitkey=compuInternalToPhys xml.sequenceOffset=80
compuPhysTo Internal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** atp.Splitkey=compuPhysToInternal xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20

▽

△

Class	CompuMethod			
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** atp.Splitkey=unit xml.sequenceOffset=30

Table B.37: CompuMethod

Class	CompuNominatorDenominator			
Note	This class represents the ability to express a polynomial either as Nominator or as Denominator.			
Base	*ARObject*			
Aggregated by	CompuRationalCoeffs.compuDenominator, CompuRationalCoeffs.compuNumerator			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
v (ordered)	Numerical	*	attr	this is the list of polynomial factors. Note that the first vf represents the power=0. The polynomial is v[0] * x^0 + v[1] * x^1 ... **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table B.38: CompuNominatorDenominator

Class	CompuRationalCoeffs			
Note	This meta-class represents the ability to express a rational function by specifying the coefficients of nominator and denominator.			
Base	*ARObject*			
Aggregated by	CompuScaleRationalFormula.compuRationalCoeffs			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
compu Denominator	CompuNominator Denominator	0..1	aggr	This is the denominator of the expression. **Tags:** xml.sequenceOffset=30
compu Numerator	CompuNominator Denominator	0..1	aggr	This is the numerator of the rational expression. **Tags:** xml.sequenceOffset=20

Table B.39: CompuRationalCoeffs

Class	CompuScale			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Aggregated by	CompuScales.compuScale			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
a2lDisplayText	String	0..1	attr	The value of this attribute shall be taken for generating one display text (specifically the OutVal) within the equivalent of the enclosing `CompuMethod` in A2L.

▽

△

Class	**CompuScale**			
compuInverse Value	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:** atp.Splitkey=compuInverseValue xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:** atp.Splitkey=desc xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveUnlimitedInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:** xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:** xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table B.40: CompuScale

Class	**CompuScaleConstantContents**
Note	This meta-class represents the fact that a particular scale of the computation method is constant.
Base	*ARObject*, *CompuScaleContents*
Aggregated by	CompuScale.compuScaleContents

▽

△

Class	CompuScaleConstantContents			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuConst	CompuConst	0..1	aggr	This represents the fact that the scale is a constant. The use case is mainly a non interpolated scale. It is a simplification of the fact that a constant scale can also be expressed as rational function of order 0. **Tags:** xml.sequenceOffset=90

Table B.41: CompuScaleConstantContents

Class	CompuScales			
Note	This meta-class represents the ability to stepwise express a computation method.			
Base	*ARObject*, *CompuContent*			
Aggregated by	Compu.compuContent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuScale (ordered)	CompuScale	*	aggr	This represents one scale within the compu method. Note that it contains a Variationpoint in order to support blueprints of enumerations. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false

Table B.42: CompuScales

Class	CpSoftwareCluster			
Note	This meta class provides the ability to define a CP Software Cluster. Each CP Software Cluster can be integrated and build individually. It defines the sub-set of hierarchical tree(s) of Software Components belonging to this CP Software Cluster. Resources required or provided by this CP Software Cluster are given in the according mappings. **Tags:** atp.recommendedPackage=CpSoftwareClusters This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
softwareCluster Id	PositiveInteger	0..1	attr	This attribute represents the value of the id of the corresponding CP software cluster.
swComponent Assignment	SwComponent PrototypeAssignment	*	aggr	This is the collection of SwComponentPrototype Assignments **Stereotypes:** atpVariation **Tags:** atp.Splitkey=swComponentAssignment, swComponent Assignment.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**CpSoftwareCluster**			
swComposition	CompositionSw ComponentType	*	ref	Software Components in the context of a CompositionSw ComponentType belonging to this CP Software Cluster. This reference can be used to describe the belonging SWCs when the CP Software Cluster is described out of the context of a System, e.g. reusable CP Software Cluster. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=swComposition, swComposition.composition SwComponentType, swComposition.variationPoint.short Label vh.latestBindingTime=systemDesignTime

Table B.43: CpSoftwareCluster

Class	**CpSoftwareClusterBinaryManifestDescriptor**			
Note	This meta-class has the ability to act as a hub for all information related to the binary manifest of a given CP software cluster. The manifest is subject to integrator work and therefore not a part of the definition of the CP software cluster itself. **Tags:** atp.recommendedPackage=CpSoftwareClusterBinaryManifestDescriptors This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
cpSoftware Cluster	CpSoftwareCluster	0..1	ref	This reference identifies the CpSoftwareCluster to which the enclosing CpSoftwareClusterBinaryManifest Descriptor belongs, The CpSoftwareClusterBinaryManifestDescriptor is defined in an integration phase while the referenced Cp SoftwareCluster represents a design element. Therefore, it makes sense to use a reference rather than an aggregation in the relation of the two meta-classes. **Tags:** atp.Splitkey=cpSoftwareCluster
metaDataField	BinaryManifestMeta DataField	*	aggr	This aggregation identifies the collection of meta-data contained in the enclosing binary manifest. **Tags:** atp.Splitkey=metaDataField, metaDataField.short Name
provide Resource	BinaryManifestProvide Resource	*	aggr	This aggregation represents the collection of provided resources in the enclosing binary manifest. **Tags:** atp.Splitkey=provideResource, provide Resource.shortName
require Resource	BinaryManifestRequire Resource	*	aggr	This aggregation represents the collection of required resources in the enclosing binary manifest. **Tags:** atp.Splitkey=requireResource, require Resource.shortName
resource Definition	BinaryManifest ResourceDefinition	*	aggr	This aggregation represents the collection of binary manifest resource definitions that belong to the enclosing CpSoftwareClusterBinaryManifestDescriptor. **Tags:** atp.Splitkey=resourceDefinition, resource Definition.shortName

▽

△

Class	CpSoftwareClusterBinaryManifestDescriptor			
softwareClusterId	PositiveInteger	0..1	attr	This attribute represents the value of the id of the corresponding CP software cluster. This id is assigned by an integrator, but may also be copied from CpSoftwareCluster.softwareClusterId if available.

Table B.44: CpSoftwareClusterBinaryManifestDescriptor

Class	CpSoftwareClusterCommunicationResource			
Note	Represents a single resource required or provided by a CP Software Cluster which relates to the port based communication on VFB level. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *CpSoftwareClusterResource*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CpSoftwareClusterResourcePool.resource			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communicationResourceProps	CpSoftwareClusterCommunicationResourceProps	0..1	aggr	This aggregation supports the further qualification of the enclosing CpSoftwareClusterCommunicationRecource by means of additional attributes depending on the nature of the CpSoftwareClusterCommunicationRecource. **Tags:** atp.Splitkey=communicationResourceProps

Table B.45: CpSoftwareClusterCommunicationResource

Class	*CpSoftwareClusterResource* (abstract)			
Note	Represents a single resource required or provided by a CP Software Cluster. **Tags:** atp.recommendedPackage=Resources This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CpSoftwareClusterCommunicationResource, CpSoftwareClusterServiceResource			
Aggregated by	CpSoftwareClusterResourcePool.resource			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dependentResource	RoleBasedResourceDependency	*	aggr	Link to a resource which depends on this resource to implement them. **Tags:** atp.Splitkey=dependentResource
globalResourceId	PositiveInteger	0..1	attr	A unique identifiers per resource used for the connection process. The identifier is required to be unique in the scope of a single machine. If software clusters are designed to be reused on multiple machines the uniqueness requirements applies for all the intended machines.
isMandatory	Boolean	0..1	attr	This attribute indicates, that the resource is mandatory to operate the Software Cluster. If the resource is not provided on the machine the connection process of any Software Cluster requiring this resource gets aborted.

Table B.46: CpSoftwareClusterResource

Class	CpSoftwareClusterResourcePool
Note	Represents the pool of resources which can be provided or required by CP Software Clusters. **Tags:** atp.recommendedPackage=CpSoftwareClusterResourcePools This Class is only used by the AUTOSAR Classic Platform.

▽

△

Class	CpSoftwareClusterResourcePool			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ecuScope	EcuInstance	*	ref	This reference identifies the EcuInstance in which the resource pool is defined. **Tags:** atp.Splitkey=ecuScope
resource	CpSoftwareClusterResource	*	aggr	This aggregation represents the collection of resources in the enclosing resource pool. **Tags:** atp.Splitkey=resource, resource.shortName

Table B.47: CpSoftwareClusterResourcePool

Class	CpSoftwareClusterServiceResource			
Note	Represents a single resource required or provided by a CP Software Cluster which relates to the BSW. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *CpSoftwareClusterResource*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CpSoftwareClusterResourcePool.resource			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
resourceNeeds	EcucContainerValue	*	ref	Refernce(s) to one or multiple EcucContainerValue(s) qualifying the characteristics of the resource. **Tags:** atp.Splitkey=resourceNeeds

Table B.48: CpSoftwareClusterServiceResource

Class	DataComProps			
Note	Represents a single resource required or provided by a CP Software Cluster which relates to the port based communication on VFB level. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *CpSoftwareClusterCommunicationResourceProps*			
Aggregated by	CpSoftwareClusterCommunicationResource.communicationResourceProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
data Consistency Policy	DataConsistencyPolicyEnum	0..1	attr	This attribute defines requirements on the data consistency mechanism in the cross cluster communication. If the attribute is not set, the default value consistencyMechanismRequired applies.
sendIndication	SendIndicationEnum	0..1	attr	Send indication behavior for last-is-the best data communication.

Table B.49: DataComProps

Enumeration	DataConsistencyPolicyEnum
Note	Defines how data consistency is ensured in the cross cluster communication.
Aggregated by	DataComProps.dataConsistencyPolicy
Literal	**Description**

▽

△

Enumeration	**DataConsistencyPolicyEnum**
consistency Mechanism Required	In this case the data consistency is ensured by the implementation of the SwClucC module. **Tags:** atp.EnumerationLiteralIndex=0
noConsistency Mechanism	In this case the data consistency is not ensured by the SwClucC module. In this case it has to be ensured by scheduling. **Tags:** atp.EnumerationLiteralIndex=1

Table B.50: DataConsistencyPolicyEnum

Class	**DataConstr**			
Note	This meta-class represents the ability to specify constraints on data. **Tags:** atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** atp.Splitkey=dataConstrRule xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table B.51: DataConstr

Class	**DataConstrRule**			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	*ARObject*			
Aggregated by	DataConstr.dataConstrRule			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. **Tags:** xml.sequenceOffset=20
internalConstrs	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). **Tags:** atp.Splitkey=internalConstrs xml.sequenceOffset=40
physConstrs	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). **Tags:** atp.Splitkey=physConstrs xml.sequenceOffset=30

Table B.52: DataConstrRule

Class	***DataMapping*** (abstract)			
Note	Mapping of port elements (data elements and parameters) to frames and signals.			
Base	*ARObject*			
Subclasses	ClientServerToSignalMapping, SenderReceiverToSignalGroupMapping, SenderReceiverToSignal Mapping, TriggerToSignalMapping			
Aggregated by	SystemMapping.dataMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
introduction	DocumentationBlock	0..1	aggr	This represents introductory documentation about the data mapping. **Tags:** atp.Splitkey=introduction

Table B.53: DataMapping

Class	***DataPrototype*** (abstract)			
Note	Base class for prototypical roles of any data type.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ApplicationCompositeElementDataPrototype*, *AutosarDataPrototype*			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level. **Tags:** atp.Splitkey=swDataDefProps

Table B.54: DataPrototype

Class	**DataReceiveErrorEvent**			
Note	This event is raised when the Com layer detects and notifies an error concerning the reception of the referenced `VariableDataPrototype`.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
data	VariableDataPrototype	0..1	iref	The referenced VariableDataPrototype raises this Data ReceiveErrorEvent when there was an error during the reception. **Tags:** atp.Splitkey=data, data.contextRPort, data.target DataElement **InstanceRef implemented by:** RVariableInAtomicSwc InstanceRef

Table B.55: DataReceiveErrorEvent

Class	**DataReceivedEvent**			
Note	This event is raised when the referenced data element is received.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***



△

Class	**DataReceivedEvent**			
data	VariableDataPrototype	0..1	iref	The referenced VariableDataPrototype raises this DataReceivedEvent when the data has been received. **Tags:** atp.Splitkey=data, data.contextRPort, data.targetDataElement **InstanceRef implemented by:** RVariableInAtomicSwcInstanceRef

Table B.56: DataReceivedEvent

Class	**DataSendCompletedEvent**			
Note	This event is raised when the referenced explicit data element has been sent or an error occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventSource	VariableAccess	0..1	ref	The referenced `VariableAccess` raises this `DataSendCompletedEvent` when the explicit write access was successful or an error occurred. **Tags:** atp.Splitkey=eventSource

Table B.57: DataSendCompletedEvent

Class	**DataWriteCompletedEvent**			
Note	This event is raised when an implicit write access was successful or an error occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventSource	VariableAccess	0..1	ref	The referenced `VariableAccess` raises this `DataWriteCompletedEvent` when the implicit write access was successful or an error occurred. **Tags:** atp.Splitkey=eventSource

Table B.58: DataWriteCompletedEvent

Class	**DiagnosticEventInfoNeeds**			
Note	This meta-class represents the needs of a software-component interested to get information regarding specific DTCs.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.

▽

△

Class	**DiagnosticEventInfoNeeds**			
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.

Table B.59: DiagnosticEventInfoNeeds

Class	**DiagnosticEventNeeds**			
Note	Specifies the abstract needs on the configuration of the Diagnostic Event Manager for one diagnostic event. Its shortName can be regarded as a symbol identifying the diagnostic event from the viewpoint of the component or module which owns this element. In case the diagnostic event specifies a production error, the shortName shall be the name of the production error.			
Base	*ARObject*, *DiagnosticCapabilityElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *Service Needs*			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deferringFid	FunctionInhibitionNeeds	*	ref	This reference contains the link to a function identifier within the FiM which is used by the monitor before delivering a result. **Tags:** atp.Splitkey=deferringFid
diagEvent Debounce Algorithm	DiagEventDebounce Algorithm	0..1	aggr	Specifies the abstract need on the Debounce Algorithm applied by the Diagnostic Event Manager. **Tags:** atp.Splitkey=diagEventDebounceAlgorithm, diag EventDebounceAlgorithm.shortName
inhibitingFid	FunctionInhibitionNeeds	0..1	ref	This represents the primary Function Inhibition Identifier used for inhibition of the diagnostic monitor. The FID might either inhibit the monitoring of a symptom or the reporting of detected faults. **Tags:** atp.Splitkey=inhibitingFid
inhibiting SecondaryFid	FunctionInhibitionNeeds	*	ref	This represents the secondary Function Inhibition Identifier used for inhibition of the diagnostic monitor. Any of the FID inhibitions leads to an inhibition of the monitoring of a symptom or the reporting of detected faults. **Tags:** atp.Splitkey=inhibitingSecondaryFid
prestored Freezeframe StoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm).
usesMonitor Data	Boolean	0..1	attr	This attribute defines whether additional monitor data shall be added to the reporting of events.

Table B.60: DiagnosticEventNeeds

Class	EcuInstance			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:** atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignal IPduGroups are applicable for which Communication Connector/ ECU. Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPdu Groups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup. **Tags:** atp.Splitkey=associatedComIPduGroup This Attribute is only used by the AUTOSAR Classic Platform.
associatedConsumedProvidedServiceInstanceGroup	ConsumedProvidedServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=associatedConsumedProvidedServiceInstanceGroup, associatedConsumedProvidedServiceInstanceGroup.consumedProvidedServiceInstanceGroup, associatedConsumedProvidedServiceInstanceGroup.variationPoint.shortLabel vh.latestBindingTime=postBuild
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU. **Tags:** atp.Splitkey=associatedPdurIPduGroup
channelSynchronousWakeup	Boolean	0..1	attr	If this parameter is available and set to true, then all available channels will be woken up as soon as at least one channel wakeup occurs. If PNCs are configured, then all PNCs will be requested upon a channel wakeup.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication. **Tags:** atp.Splitkey=clientIdRange
comConfigurationGwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRouteSignals of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
comConfigurationRxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRx of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
comConfigurationTxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionTx of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.



△

Class	EcuInstance			
comEnable MDTForCyclic Transmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclic Timing assigned or eventControlledTiming with numberOf Repetitions > 0). This Attribute is only used by the AUTOSAR Classic Platform.
commController	Communication Controller	*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=commController, commController.short Name, commController.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	Communication Connector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
dltConfig	DltConfig	0..1	aggr	Describes the Dlt configuration on this EcuInstance. **Tags:** atp.Splitkey=dltConfig This Attribute is only used by the AUTOSAR Classic Platform.
doIpConfig	DoIpConfig	0..1	aggr	DoIp configuration on this EcuInstance. **Tags:** atp.Splitkey=doIpConfig atp.Status=draft This Attribute is only used by the AUTOSAR Classic Platform.
ecuTaskProxy	OsTaskProxy	*	ref	Reference to OsTaskProxies assigned to the Ecu Instance. **Tags:** atp.Splitkey=ecuTaskProxy This Attribute is only used by the AUTOSAR Classic Platform.
ethSwitchPort Group Derivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done. This Attribute is only used by the AUTOSAR Classic Platform.
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of an EcuInstance. **Tags:** atp.Splitkey=firewallRule
j1939Node	J1939Node	*	aggr	Optional collection of J1939Nodes defined on this Ecu Instance. **Tags:** atp.Splitkey=j1939Node, j1939Node.shortName This Attribute is only used by the AUTOSAR Classic Platform.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu. **Tags:** atp.Splitkey=partition, partition.shortName This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	EcuInstance			
pncNmRequest	Boolean	0..1	attr	Defines if this EcuInstance shall request Nm on all its PhysicalChannels which have Nm variant set to FULL each time a PNC is requested.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
sleepMode Supported	Boolean	0..1	attr	Specifies whether the ECU instance may be put to a "low power mode" • true: sleep mode is supported • false: sleep mode is not supported Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software. This Attribute is only used by the AUTOSAR Classic Platform.
tcpIpIcmpProps	EthTcpIpIcmpProps	0..1	ref	EcuInstance specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.Splitkey=tcpIpIcmpProps This Attribute is only used by the AUTOSAR Classic Platform.
tcpIpProps	EthTcpIpProps	0..1	ref	EcuInstance specific TcpIp Stack attributes. **Tags:** atp.Splitkey=tcpIpProps This Attribute is only used by the AUTOSAR Classic Platform.
wakeUpOver BusSupported	Boolean	0..1	attr	Driver support for wakeup over Bus. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.61: EcuInstance

Class	*EcucContainerDef* (abstract)			
Note	Base class used to gather common attributes of configuration container definitions. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucChoiceContainerDef, EcucParamConfContainerDef			
Aggregated by	EcucDestinationUriPolicy.container, EcucModuleDef.container, EcucParamConfContainerDef.sub Container			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
destinationUri	EcucDestinationUriDef	*	ref	Several destinationUris can be defined for an Ecuc ContainerDef. With such destinationUris an Ecuc ContainerDef is applicable for several EcucUriReference Defs. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=destinationUri

▽

△

Class	***EcucContainerDef*** (abstract)			
linkTime Support	Boolean	0..1	attr	Indicates whether the container multiplicity can be changed at LinkTime. Please note that Link-Time is only supported in the Vendor Specific Module Definition (VSMD).
multiplicity ConfigClass	EcucMultiplicity ConfigurationClass	*	aggr	Specifies which MultiplicityConfigurationClass this container is available for which ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_ DEFINITION. If the category attribute of the EcucModule Def is set to VENDOR_SPECIFIC_MODULE_ DEFINITION and if the upperMultiplicity is greater than the lowerMultiplicity then this aggregation is mandatory. **Tags:** atp.Splitkey=multiplicityConfigClass atp.Status=obsolete xml.namePlural=MULTIPLICITY-CONFIG-CLASSES
origin	String	0..1	attr	This attribute specifies whether this configuration container is an AUTOSAR standardized container or whether it is vendor-specific.
postBuild Loadable Multiplicity	Boolean	0..1	attr	Indicates whether the container multiplicity can be changed at post-build loadable time by updating the configuration binary. TRUE means yes, FALSE means no.
postBuildVariant Multiplicity	Boolean	0..1	attr	Indicates if a container may have different number of instances in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
requiresIndex	Boolean	0..1	attr	Used to define whether the value element for this definition shall be provided with an index.

Table B.62: EcucContainerDef

Class	**EcucContainerValue**			
Note	Represents a Container definition in the ECU Configuration Description. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucContainerValue.subContainer, EcucModuleConfigurationValues.container			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
definition	EcucContainerDef	0..1	ref	Reference to the definition of this Container in the ECU Configuration Parameter Definition. **Tags:** atp.Splitkey=definition xml.sequenceOffset=-10
parameterValue	EcucParameterValue	*	aggr	Aggregates all ECU Configuration Values within this Container. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=parameterValue, parameterValue.variation Point.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	EcucContainerValue			
referenceValue	EcucAbstractReference Value	*	aggr	Aggregates all References with this container. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=referenceValue, referenceValue.variation Point.shortLabel vh.latestBindingTime=postBuild
subContainer	EcucContainerValue	*	aggr	Aggregates all sub-containers within this container. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=subContainer, subContainer.shortName, sub Container.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table B.63: EcucContainerValue

Class	EcucEnumerationParamDef			
Note	Configuration parameter type for Enumeration. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
defaultValue	Identifier	0..1	attr	Default value of the enumeration configuration parameter. This string needs to be one of the literals specified for this enumeration.
literal	EcucEnumerationLiteral Def	*	aggr	Aggregation on the literals used to define this enumeration parameter. This aggregation is optional if the surrounding EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION then this aggregation is mandatory. **Tags:** atp.Splitkey=literal, literal.shortName

Table B.64: EcucEnumerationParamDef

Class	EcucForeignReferenceDef			
Note	Specify a reference to an XML description of an entity described in another AUTOSAR template. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractExternalReferenceDef*, *EcucAbstractReferenceDef*, *Ecuc CommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationType	String	0..1	attr	The type in the AUTOSAR Metamodel to which instance this reference is allowed to point to.

Table B.65: EcucForeignReferenceDef

Class	**EcucModuleConfigurationValues**			
Note	Head of the configuration of one Module. A Module can be a BSW module as well as the RTE and ECU Infrastructure. As part of the BSW module description, the EcucModuleConfigurationValues element has two different roles: The recommendedConfiguration contains parameter values recommended by the BSW module vendor. The preconfiguredConfiguration contains values for those parameters which are fixed by the implementation and cannot be changed. These two EcucModuleConfigurationValues are used when the base EcucModuleConfigurationValues (as part of the base ECU configuration) is created to fill parameters with initial values. **Tags:** atp.recommendedPackage=EcucModuleConfigurationValuess This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
container	EcucContainerValue	*	aggr	Aggregates all containers that belong to this module configuration. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=container, container.shortName, container.variationPoint.shortLabel vh.latestBindingTime=postBuild xml.sequenceOffset=10
definition	EcucModuleDef	0..1	ref	Reference to the definition of this EcucModule ConfigurationValues element. Typically, this is a vendor specific module configuration. **Tags:** atp.Splitkey=definition xml.sequenceOffset=-10
ecucDefEdition	RevisionLabelString	0..1	attr	This is the version info of the ModuleDef ECUC Parameter definition to which this values conform to / are based on. For the Definition of ModuleDef ECUC Parameters the AdminData shall be used to express the semantic changes. The compatibility rules between the definition and value revision labels is up to the module's vendor.
implementation ConfigVariant	EcucConfiguration VariantEnum	0..1	attr	Specifies the kind of deliverable this EcucModule ConfigurationValues element provides. If this element is not used in a particular role (e.g. preconfigured Configuration or recommendedConfiguration) then the value shall be one of VariantPreCompile, VariantLink Time, VariantPostBuild. **Tags:** atp.Status=obsolete
module Description	BswImplementation	0..1	ref	Referencing the BSW module description, which this EcucModuleConfigurationValues element is configuring. This is optional because the EcucModuleConfiguration Values element is also used to configure the ECU infrastructure (memory map) or Application SW-Cs. However in case the EcucModuleConfigurationValues are used to configure the module, the reference is mandatory in order to fetch module specific "common" published information. **Tags:** atp.Splitkey=moduleDescription

▽

△

Class	EcucModuleConfigurationValues			
postBuildVariant Used	Boolean	0..1	attr	Indicates whether a module implementation has or plans to have (i.e., introduced at link or post-build time) new post-build variation points. TRUE means yes, FALSE means no. If the attribute is not defined, FALSE semantics shall be assumed.

Table B.66: EcucModuleConfigurationValues

Class	EcucModuleDef			
Note	Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure. **Tags:** atp.recommendedPackage=EcucDefs This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpDefinition*, *CollectableElement*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
apiServicePrefix	CIdentifier	0..1	attr	For modules where several instances of the VSMD can be defined the apiServicePrefix defines the API namespace of the derived instances, e.g. Cdd, Xfrm (ComXf, SomeIpXf, E2EXf).
container	EcucContainerDef	*	aggr	Aggregates the top-level container definitions of this specific module definition. **Tags:** atp.Splitkey=container, container.shortName xml.sequenceOffset=11
postBuild Loadable Support	Boolean	0..1	attr	Indicates if the module supports changing the configuration by loading a new configuration binary at postBuild time. TRUE means yes, FALSE means no.
postBuildVariant Support	Boolean	0..1	attr	Indicates if a module supports different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
refinedModule Def	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_ DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=refinedModuleDef
supported ConfigVariant	EcucConfiguration VariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the Ecuc ModuleDef has the category STANDARDIZED_ MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_ MODULE_DEFINITION then this attribute is mandatory. **Tags:** atp.Status=obsolete

Table B.67: EcucModuleDef

Class	EcucParamConfContainerDef			
Note	Used to define configuration containers that can hierarchically contain other containers and/or parameter definitions. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucContainerDef*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucChoiceContainerDef.choice, EcucDestinationUriPolicy.container, EcucModuleDef.container, EcucParamConfContainerDef.subContainer			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
parameter	EcucParameterDef	*	aggr	The parameters defined within the EcucParamConfContainerDef. **Tags:** atp.Splitkey=parameter, parameter.shortName
reference	EcucAbstractReferenceDef	*	aggr	The references defined within the EcucParamConfContainerDef. **Tags:** atp.Splitkey=reference, reference.shortName
subContainer	EcucContainerDef	*	aggr	The containers defined within the EcucParamConfContainerDef. **Tags:** atp.Splitkey=subContainer, subContainer.shortName

Table B.68: EcucParamConfContainerDef

Class	**ExecutableEntity** (abstract)			
Note	Abstraction of executable code.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*BswModuleEntity*, RunnableEntity			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activationReason	ExecutableEntityActivationReason	*	aggr	If the `ExecutableEntity` provides at least one `activationReason` element the RTE resp. BSW Scheduler shall provide means to read the activation vector of this executable entity execution. If no `activationReason` element is provided the feature of being able to determine the activating RTEEvent is disabled for this `ExecutableEntity`. **Tags:** atp.Splitkey=activationReason, activationReason.shortName
canEnter	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=canEnter, canEnter.exclusiveArea, canEnter.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveAreaNestingOrder	ExclusiveAreaNestingOrder	*	ref	This represents the set of ExclusiveAreaNestingOrders recognized by this ExecutableEntity. **Tags:** atp.Splitkey=exclusiveAreaNestingOrder
minimumStartInterval	TimeValue	0..1	attr	Specifies the time in seconds by which two consecutive starts of an ExecutableEntity are guaranteed to be separated.



△

Class	***ExecutableEntity*** (abstract)			
reentrancyLevel	ReentrancyLevelEnum	0..1	attr	The reentrancy level of this ExecutableEntity. See the documentation of the enumeration type ReentrancyLevel Enum for details. Please note that nonReentrant interfaces can have also reentrant or multicoreReentrant implementations, and reentrant interfaces can also have multicoreReentrant implementations.
runsInside	ExclusiveArea	*	ref	The executable entity runs completely inside the referenced exclusive area. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=runsInside, runsInside.exclusiveArea, runsInside.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this code entity. Via an association to the same SwAddrMethod, it can be specified that several code entities (even of different modules or components) shall be located in the same memory without already specifying the memory section itself. **Tags:** atp.Splitkey=swAddrMethod

Table B.69: ExecutableEntity

Class	**ExternalTriggerOccurredEvent**			
Note	This event is raised when the referenced Trigger has occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
trigger	Trigger	0..1	iref	The referenced Trigger raises this ExternalTriggerOccurredEvent. **Tags:** atp.Splitkey=trigger, trigger.contextRPort, trigger.targetTrigger **InstanceRef implemented by:** RTriggerInAtomicSwcInstanceRef

Table B.70: ExternalTriggerOccurredEvent

Class	**ExternalTriggeringPoint**			
Note	If a RunnableEntity owns an ExternalTriggeringPoint it is entitled to raise an ExternalTriggerOccurred Event.			
Base	*ARObject*			
Aggregated by	RunnableEntity.externalTriggeringPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ExternalTriggeringPoint**			
ident	ExternalTriggeringPoint Ident	0..1	aggr	The aggregation in the role ident provides the ability to make the ExternalTriggeringPoint identifiable. From the semantical point of view, the ExternalTriggering Point is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let ModeAccessPoint directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor **Tags:** xml.sequenceOffset=-100
trigger	Trigger	0..1	iref	The trigger taken for the ExternalTriggeringPoint. **Tags:** atp.Splitkey=trigger, trigger.contextPPort, trigger.target Trigger xml.namePlural=TRIGGER-IREF xml.roleElement=false xml.roleWrapperElement=true xml.typeElement=true xml.typeWrapperElement=false **InstanceRef implemented by:** PTriggerInAtomicSwc TypeInstanceRef

Table B.71: ExternalTriggeringPoint

Class	**FlatInstanceDescriptor**			
Note	Represents exactly one node (e.g. a component instance or data element) of the instance tree of a software system. The purpose of this element is to map the various nested representations of this instance to a flat representation and assign a unique name (shortName) to it. Use cases: • Specify unique names of measurable data to be used by MCD tools • Specify unique names of calibration data to be used by MCD tool • Specify a unique name for an instance of a component prototype in the ECU extract of the system description Note that in addition it is possible to assign alias names via AliasNameAssignment. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	FlatMap.instance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bsw Implementation	BswImplementation	0..1	ref	Reference to BswImplementation that defines the context for the AutosarDataPrototype that is referenced by Flat InstanceDescriptor.dataPrototype. **Tags:** atp.Splitkey=bswImplementation
dataPrototype	AutosarDataPrototype	0..1	ref	Reference to a DataPrototype that is defined in the Bsw InternalBehavior in the context of a BswImplementation that is defined by the FlatInstanceDescriptor.bsw Implementation reference. **Tags:** atp.Splitkey=dataPrototype

▽

△

Class	**FlatInstanceDescriptor**			
ecuExtract Reference	AtpFeature	0..1	iref	Refers to the instance in the ECU extract. This is valid only, if the FlatMap is used in the context of an ECU extract. The reference shall be such that it uniquely defines the object instance. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying instance of the component prototype and the Atomic SoftwareComponentType, which is refered by the particular SwcInternalBehavior. **Tags:** atp.Splitkey=ecuExtractReference, ecuExtract Reference.contextElement, ecuExtractReference.target xml.sequenceOffset=40 **InstanceRef implemented by:** AnyInstanceRef
role	Identifier	0..1	attr	The role denotes the particular role of the downstream memory location described by this FlatInstanceDescriptor. It applies to use case where one upstream object results in multiple downstream objects, e.g. ModeDeclaration GroupPrototypes which are measurable. In this case the RTE will provide locations for current mode, previous mode and next mode.
rtePluginProps	RtePluginProps	0..1	aggr	The properties of a communication graph with respect to the utilization of RTE Implementation Plug-in. **Tags:** atp.Splitkey=rtePluginProps
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this FlatInstanceDescriptor. **Tags:** atp.Splitkey=swDataDefProps
upstream Reference	AtpFeature	0..1	iref	Refers to the instance in the context of an "upstream" description, which could be: the SYSTEM_ DESCRIPTION, or SYSTEM_EXTRACT, or ECU_ SYSTEM_DESCRIPTION, or SW_CLUSTER_SYSTEM_ DESCRIPTION, or the basic software module description (in this case only the target reference of the AnyInstance Ref is needed), or (if a flat map is used in preliminary context) a description of an atomic component or composition. This reference is optional in case the flat map is used in ECU context. The reference shall be such that it uniquely defines the object instance in the given context. For example, if a data prototype is declared as a role within an SwcInternal Behavior, it is not enough to state the Swc Internal Behavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying the instance of the component prototype that contains the particular instance of Swc InternalBehavior. **Tags:** atp.Splitkey=upstreamReference, upstream Reference.contextElement, upstreamReference.target xml.sequenceOffset=20 **InstanceRef implemented by:** AnyInstanceRef

Table B.72: FlatInstanceDescriptor

Class	**FunctionInhibitionAvailabilityNeeds**			
Note	Specifies the abstract needs on the configuration of the Function Inhibition Manager to provide the control function for one Function Identifier (FID).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
controlledFid	FunctionInhibitionNeeds	0..1	ref	This reference represents the controlled FID **Tags:** atp.Splitkey=controlledFid

Table B.73: FunctionInhibitionAvailabilityNeeds

Class	**FunctionInhibitionNeeds**			
Note	Specifies the abstract needs on the configuration of the Function Inhibition Manager for one Function Identifier (FID). This class currently contains no attributes. Its name can be regarded as a symbol identifying the FID from the viewpoint of the component or module which owns this class.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.74: FunctionInhibitionNeeds

Enumeration	**HandleInvalidEnum**
Note	Strategies of handling the reception of invalidValue.
Aggregated by	InvalidationPolicy.handleInvalid, ISignalPort.handleInvalid
Literal	***Description***
dontInvalidate	Invalidation is switched off. **Tags:** atp.EnumerationLiteralIndex=0
external Replacement	Replace a received invalidValue. The replacement value is sourced from the aggregation in the role replaceWith. **Tags:** atp.EnumerationLiteralIndex=1
keep	The application software is supposed to handle signal invalidation on RTE API level either by Data ReceiveErrorEvent or check of error code on read access. **Tags:** atp.EnumerationLiteralIndex=2
replace	Replace a received invalidValue. The replacement value is specified by the initValue. **Tags:** atp.EnumerationLiteralIndex=3

Table B.75: HandleInvalidEnum

Class	***Identifiable*** (abstract)
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*



△

Class	**Identifiable** (abstract)			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, AppOsTask ProxyToEcuTaskProxyMapping, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartition Mapping, AppliedStandard, AsynchronousServerCallResultPoint, *AtpBlueprint*, *AtpClassifier*, *Atp Feature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifestAddressable Object*, BinaryManifestItemDefinition, *BinaryManifestResource*, BinaryManifestResourceDefinition, Block State, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, BuildActionEnvironment, CanSecSecureChannel, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClientIdDefinition, Client ServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEvent Group, *CouplingElementAbstractDetails*, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPort StructuralElement*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplicationPartition Mapping, CpSoftwareClusterToApplicationPartitionMapping, CpSoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, *CryptoServiceMapping*, CyclicHandlingComDataToOsTask ProxyMapping, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, DataTransformation, *Dds AbstractServiceInstanceElementCp*, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic ExtendedDataRecordElement, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DltApplication, DltArgument, DltArgumentProps, DltLogChannel, Dlt Message, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, ECUMapping, *EOCExecutableEntity RefAbstract*, EcuPartition, EcuPartitionToCoreMapping, EcucContainerValue, *EcucDefinitionElement*, EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, Ethernet WakeupSleepOnDatalineConfig, EventHandler, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeature Relation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayTpConnection Control, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722 TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignal Triggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, J1939Node, J1939SharedAddress Cluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacAddress VlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, Mode Declaration, ModeDeclarationMapping, ModeSwitchPoint, ModeSwitchSenderComSpecProps, Network Endpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *Physical Channel*, PortElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, Queued ReceiverComSpecProps, ResourceConsumption, RootSwCompositionPrototype, RptComponent, Rpt Container, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptService Point, RteEventInCompositionSeparation, RteEventInCompositionToOsTaskProxyMapping, RteEventIn SystemSeparation, RteEventInSystemToOsTaskProxyMapping, RunnableEntityGroup, *SdgAttribute*, Sdg Class, SecOcJobRequirement, SecureCommunicationAuthenticationProps, SecureCommunication FreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServerCallPoint*, ServerComSpecProps, *ServiceNeeds*, SignalServiceTranslationElementProps, SignalServiceTranslation EventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, *StackUsage*, Static SocketConnection, StructuredReq, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SwitchAsynchronousTraffic ShaperGroupEntry, SwitchAtsInstanceEntry, SwitchFlowMeteringEntry, SwitchStreamFilterActionDest PortModification, SwitchStreamFilterEntry, SwitchStreamFilterRule, SwitchStreamGateEntry, Switch StreamIdentification, SystemMapping, SystemSignalGroupToCommunicationResourceMapping, System SignalToCommunicationResourceMapping, TDCpSoftwareClusterMapping, TDCpSoftwareCluster ResourceMapping, TcpOptionFilterList, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipher Suite, TlsCryptoCipherSuiteProps, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, Variable Access, VariationPointProxy, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40

▽

△

Class	***Identifiable*** (abstract)			
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** atp.Splitkey=annotation xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** atp.Splitkey=desc xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** atp.Splitkey=introduction xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table B.76: Identifiable

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

▽

△

Class	**ImplementationDataType**			
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 B.77: ImplementationDataType

Class	**ImplementationDataTypeElement**			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further `subElement`s or it is further defined via its `swDataDefProps`. There are several use cases within the system of `ImplementationDataType`s for such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ImplementationDataType.subElement, ImplementationDataTypeElement.sub Element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arrayImplPolicy	ArrayImplPolicyEnum	0..1	attr	This attribute controls the implementation of the payload of an array. It shall only be used if the enclosing `ImplementationDataType` constitutes an array.

▽

△

Class	**ImplementationDataTypeElement**			
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this `ImplementationDataTypeElement` represents the type of each single array element. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing `ImplementationDataTypeElement` as optional. This means that, at runtime, the `ImplementationDataTypeElement` may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the `CppImplementationDataTypeElement` as not valid at the sending end of a communication and determine its validity at the receiving end.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). 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
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement. **Tags:** atp.Splitkey=swDataDefProps

Table B.78: ImplementationDataTypeElement

Class	**InitEvent**			
Note	This `RTEEvent` is supposed to be used for initialization purposes, i.e. for starting and restarting a partition. It is not guaranteed that all `RunnableEntity`s referenced by this `InitEvent` are executed before the 'regular' `RunnableEntity`s are executed for the first time. The execution order depends on the task mapping.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.79: InitEvent

Class	***InternalBehavior*** (abstract)
Note	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	BswInternalBehavior, SwcInternalBehavior

▽

△

Class	***InternalBehavior*** (abstract)			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
constant Memory	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) implemented by this Internal Behavior. The shortName of ParameterDataPrototype has to be equal to the ''C' identifier of the described constant. The characteristic value(s) might be shared between Sw ComponentPrototypes of the same SwComponentType. The aggregation of constantMemory is subject to variability with the purpose to support variability in the software component or module implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=constantMemory, constantMemory.short Name, constantMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular InternalBehavior **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular InternalBehavior **Tags:** atp.Splitkey=dataTypeMapping
exclusiveArea	ExclusiveArea	*	aggr	This specifies an ExclusiveArea for this InternalBehavior. The exclusiveArea is local to the component resp. module. The aggregation of ExclusiveAreas is subject to variability. Note: the number of ExclusiveAreas might vary due to the conditional existence of RunnableEntities or BswModuleEntities. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=exclusiveArea, exclusiveArea.shortName, exclusiveArea.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveArea NestingOrder	ExclusiveAreaNesting Order	*	aggr	This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=exclusiveAreaNestingOrder, exclusiveArea NestingOrder.shortName, exclusiveAreaNesting Order.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	***InternalBehavior*** (abstract)			
staticMemory	VariableDataPrototype	*	aggr	Describes a read and writeable static memory object representing measurerment variables implemented by this software component. The term "static" is used in the meaning of "non-temporary" and does not necessarily specify a linker encapsulation. This kind of memory is only supported if supportsMultipleInstantiation is FALSE. The shortName of the VariableDataPrototype has to be equal with the ''C' identifier of the described variable. The aggregation of staticMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=staticMemory, staticMemory.shortName, staticMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.80: InternalBehavior

Class	**InternalConstrs**			
Note	This meta-class represents the ability to express internal constraints.			
Base	*ARObject*			
Aggregated by	DataConstrRule.internalConstrs			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. **Tags:** xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in maps and curves. **Tags:** xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This element specifies the monotony characteristics of the current internal or physical limits. The following table shows the monotony characteristics which are to be filled through the corresponding values. If the element has no contents or if it is omitted, "no Monotony" is the default content. **Tags:** xml.sequenceOffset=70
upperLimit	Limit	0..1	attr	This specifies the upper limit defined by the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table B.81: InternalConstrs

Class	InternalTriggerOccurredEvent			
Note	This event is raised when the referenced `InternalTriggeringPoint` has occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventSource	InternalTriggeringPoint	0..1	ref	The referenced `InternalTriggeringPoint` raises this `InternalTriggerOccurredEvent`. **Tags:** atp.Splitkey=eventSource

Table B.82: InternalTriggerOccurredEvent

Class	InvalidationPolicy			
Note	Specifies whether the component can actively invalidate a particular `dataElement`. If no `invalidationPolicy` points to a `dataElement` this is considered to yield the identical result as if the `handleInvalid` attribute was set to `dontInvalidate`.			
Base	*ARObject*			
Aggregated by	SenderReceiverInterface.invalidationPolicy			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	VariableDataPrototype	0..1	ref	Reference to the `dataElement` for which the `InvalidationPolicy` applies. **Tags:** atp.Splitkey=dataElement
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute controls how invalidation is applied to the dataElement.

Table B.83: InvalidationPolicy

Class	McDataInstance			
Note	Describes the specific properties of one data instance in order to support measurement and/or calibration of this data instance. The most important attributes are: • Its shortName is copied from the ECU Flat map (if applicable) and will be used as identifier and for display by the MC system. • The category is copied from the corresponding data type (ApplicationDataType if defined, otherwise ImplementationDataType) as far as applicable. • The symbol is the one used in the programming language. It will be used to find out the actual memory address by the final generation tool with the help of linker generated information. It is assumed that in the M1 model this part and all the aggregated and referred elements (with the exception of the Flat Map and the references from ImplementationElementInParameterInstanceRef and McAccessDetails) are completely generated from "upstream" information. This means, that even if an element like e.g. a CompuMethod is only used via reference here, it will be copied into the M1 artifact which holds the complete McSupportData for a given Implementation.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	McDataInstance.subElement, McSupportData.mcParameterInstance, McSupportData.mcVariableInstance			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arraySize	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of number of elements.
displayIdentifier	McdIdentifier	0..1	attr	An optional attribute to be used to set the ASAM ASAP2 DISPLAY_IDENTIFIER attribute.

△

Class	**McDataInstance**			
flatMapEntry	FlatInstanceDescriptor	0..1	ref	Reference to the corresponding entry in the ECU Flat Map. This allows to trace back to the original specification of the generated data instance. This link shall be added by the RTE generator mainly for documentation purposes. The reference is optional because • The McDataInstance may represent an array or struct in which only the subElements correspond to FlatMap entries. • The McDataInstance may represent a task local buffer for rapid prototyping access which is different from the "main instance" used for measurement access. **Tags:** atp.Splitkey=flatMapEntry This Attribute is only used by the AUTOSAR Classic Platform.
instanceIn Memory	ImplementationElement InParameterInstance Ref	0..1	aggr	Reference to the corresponding data instance in the description of calibration data structures published by the RTE generator. This is used to support emulation methods inside the ECU, it is not required for A2L generation. **Tags:** atp.Splitkey=instanceInMemory, instanceIn Memory.context, instanceInMemory.target
mcDataAccess Details	McDataAccessDetails	0..1	aggr	Refers to "upstream" information on how the RTE uses this data instance. Use Case: Rapid Prototyping **Tags:** atp.Splitkey=mcDataAccessDetails
mcData Assignment	RoleBasedMcData Assignment	*	aggr	An assignment between McDataInstances. This supports the indication of related McDataElement implementing the of "RP global buffer", "RP global measurement buffer", "RP enabler flag". **Tags:** atp.Splitkey=mcDataAssignment
resulting Properties	SwDataDefProps	0..1	aggr	These are the generated properties resulting from decisions taken by the RTE generator for the actually implemented data instance. Only those properties are relevant here, which are needed for the measurement and calibration system. **Tags:** atp.Splitkey=resultingProperties
resultingRptSw Prototyping Access	RptSwPrototyping Access	0..1	aggr	Describes the implemented accessibility of data and modes by the rapid prototyping tooling. **Tags:** atp.Splitkey=resultingRptSwPrototypingAccess atp.Status=obsolete
role	Identifier	0..1	attr	An optional attribute to be used for additional information on the role of this data instance, for example in the context of rapid prototyping.
rptImplPolicy	RptImplPolicy	0..1	aggr	Describes the implemented code preparation for rapid prototyping at data accesses for a hook based bypassing. **Tags:** atp.Splitkey=rptImplPolicy atp.Status=obsolete
subElement (ordered)	McDataInstance	*	aggr	This relation indicates, that the target element is part of a "struct" which is given by the source element. This information will be used by the final generator to set up the correct addressing scheme. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

▽

△

Class	McDataInstance			
symbol	SymbolString	0..1	attr	This String is used to determine the memory address during final generation of the MC configuration data (e.g. "A2L" file) . It shall be the name of the element in the programming language such that it can be identified in linker generated information. In case the McDataInstance is part of composite data in the programming language, the symbol String may include parts denoting the element context, unless the context is given by the symbol attribute of an enclosing McDataInstance. This means in particular for the C language that the "." character shall be used as a separator between the name of a "struct" variable the name of one of its elements. The symbol can differ from the shortName in case of generated C data declarations. It is an optional attribute since it may be missing in case the instance represents an element (e.g. a single array element) which has no name in the linker map.

Table B.84: McDataInstance

Class	McSupportData			
Note	Root element for all measurement and calibration support data related to one Implementation artifact on an ECU. There shall be one such element related to the RTE implementation (if it owns MC data) and a separate one for each module or component, which owns private MC data.			
Base	*ARObject*			
Aggregated by	*Implementation*.mcSupport			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
emulation Support	McSwEmulationMethod Support	*	aggr	Describes the calibration method used by the RTE. This information is not needed for A2L generation, but to setup software emulation in the ECU. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=emulationSupport, emulation Support.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
mcParameter Instance	McDataInstance	*	aggr	A data instance to be used for calibration. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=mcParameterInstance, mcParameter Instance.shortName, mcParameterInstance.variation Point.shortLabel vh.latestBindingTime=postBuild
mcVariable Instance	McDataInstance	*	aggr	A data instance to be used for measurement. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=mcVariableInstance, mcVariable Instance.shortName, mcVariableInstance.variation Point.shortLabel vh.latestBindingTime=postBuild
measurable System ConstantValues	SwSystemconstant ValueSet	*	ref	Sets of system constant values to be transferred to the MCD system, because the system constants have been specified with "swCalibrationAccess" = readonly. **Tags:** atp.Splitkey=measurableSystemConstantValues

▽

△

Class	McSupportData			
rptSupportData	RptSupportData	0..1	aggr	The rapid prototyping support data belonging to this implementation. The aggregtion is <<atpSplitable>> because in case of an already exisiting BSW Implementation model, this description will be added later in the process, namely at code generation time. **Tags:** atp.Splitkey=rptSupportData

Table B.85: McSupportData

Class	ModeAccessPoint			
Note	A ModeAccessPoint is required by a RunnableEntity owned by a Mode Manager or Mode User. Its semantics implies the ability to access the current mode (provided by the RTE) of a ModeDeclaration GroupPrototype's ModeDeclarationGroup.			
Base	*ARObject*			
Aggregated by	RunnableEntity.modeAccessPoint			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ident	ModeAccessPointIdent	0..1	aggr	The aggregation in the role ident provides the ability to make the ModeAccessPoint identifiable. From the semantical point of view, the ModeAccessPoint is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let ModeAccessPoint directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor **Tags:** xml.sequenceOffset=-100
modeGroup	ModeDeclarationGroup Prototype	0..1	iref	The mode declaration group that is accessed by this runnable. **Tags:** atp.Splitkey=modeGroup, modeGroup.contextPPort, modeGroup.contextRPort, modeGroup.targetModeGroup xml.typeElement=true **InstanceRef implemented by:** ModeGroupInAtomicSwc InstanceRef

Table B.86: ModeAccessPoint

Class	ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ModeDeclarationGroup.modeDeclaration			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

Table B.87: ModeDeclaration

Class	ModeDeclarationGroup			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred. **Tags:** atp.Splitkey=initialMode
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=modeDeclaration, modeDeclaration.short Name, modeDeclaration.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
modeManager ErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user). **Tags:** atp.Splitkey=modeManagerErrorBehavior This Attribute is only used by the AUTOSAR Classic Platform.
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup **Tags:** atp.Splitkey=modeTransition, modeTransition.short Name This Attribute is only used by the AUTOSAR Classic Platform.
modeUserError Behavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager). **Tags:** atp.Splitkey=modeUserErrorBehavior This Attribute is only used by the AUTOSAR Classic Platform.
onTransition Value	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.88: ModeDeclarationGroup

Class	ModeDeclarationGroupPrototype			
Note	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.providedModeGroup, BswModuleDescription.requiredModeGroup, FirewallStateSwitchInterface.firewallStateMachine, FunctionGroupSet.functionGroup, ModeSwitchInterface.modeGroup, Process.processStateMachine, StateManagementStateNotification.stateMachine			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	ModeDeclarationGroupPrototype			
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	This allows for specifying whether or not the enclosing ModeDeclarationGroupPrototype can be measured at run-time. This Attribute is only used by the AUTOSAR Classic Platform.
type	ModeDeclarationGroup	0..1	tref	The "collection of ModeDeclarations" (= ModeDeclaration Group) supported by a component **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table B.89: ModeDeclarationGroupPrototype

Class	ModeDeclarationMappingSet			
Note	This meta-class implements a container for `ModeDeclarationMappings` **Tags:** atp.recommendedPackage=PortInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
mode Declaration Mapping	ModeDeclaration Mapping	*	aggr	This represents the collection of ModeDeclaration Mappings owned by the enclosing ModeDeclaration MappingSet. **Tags:** atp.Splitkey=modeDeclarationMapping, mode DeclarationMapping.shortName

Table B.90: ModeDeclarationMappingSet

Class	ModeErrorBehavior			
Note	This represents the ability to define the error behavior in the context of mode handling.			
Base	*ARObject*			
Aggregated by	ModeDeclarationGroup.modeManagerErrorBehavior, ModeDeclarationGroup.modeUserErrorBehavior			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
defaultMode	ModeDeclaration	0..1	ref	This represents the ModeDeclaration that is considered the error mode in the context of the enclosing Mode DeclarationGroup. **Tags:** atp.Splitkey=defaultMode
errorReaction Policy	ModeErrorReaction PolicyEnum	0..1	attr	This represents the ability to define the policy in terms of which default model shall apply in case an error occurs.

Table B.91: ModeErrorBehavior

Class	ModeSwitchInterface			
Note	A mode switch interface declares a `ModeDeclarationGroupPrototype` to be sent and received. **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**

▽

△

Class	ModeSwitchInterface			
modeGroup	ModeDeclarationGroup Prototype	0..1	aggr	The `ModeDeclarationGroupPrototype` of this mode interface. **Tags:** atp.Splitkey=modeGroup, modeGroup.shortName

Table B.92: ModeSwitchInterface

Class	ModeSwitchPoint			
Note	A `ModeSwitchPoint` is required by a `RunnableEntity` owned a Mode Manager. Its semantics implies the ability to initiate a mode switch.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, RunnableEntity.modeSwitchPoint			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
modeGroup	ModeDeclarationGroup Prototype	0..1	iref	The mode declaration group that is switched by this runnable. **Tags:** atp.Splitkey=modeGroup, modeGroup.context PPort, modeGroup.targetModeGroup **InstanceRef implemented by:** PModeGroupInAtomic SwcInstanceRef

Table B.93: ModeSwitchPoint

Class	ModeTransition			
Note	This meta-class represents the ability to describe possible ModeTransitions in the context of a Mode DeclarationGroup.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ModeDeclarationGroup.modeTransition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
enteredMode	ModeDeclaration	0..1	ref	This represents the entered model of the ModeTransition. **Tags:** atp.Splitkey=enteredMode
exitedMode	ModeDeclaration	0..1	ref	This represents the exited mode of the ModeTransition **Tags:** atp.Splitkey=exitedMode

Table B.94: ModeTransition

Enumeration	MonotonyEnum
Note	This enumerator denotes the values for specification of monotony for e.g. curves.
Aggregated by	InternalConstrs.monotony, PhysConstrs.monotony, *SwCalprmAxisTypeProps*.monotony
Literal	**Description**
decreasing	This indicates that the related curve needs to be monotony decreasing. **Tags:** atp.EnumerationLiteralIndex=0
increasing	This indicates that the related curve needs to be monotony increasing. **Tags:** atp.EnumerationLiteralIndex=1
monotonous	This indicates that the values shall be monotonously decreasing or increasing, depending on the trend set by the first values of the series. **Tags:** atp.EnumerationLiteralIndex=2

▽

△

Enumeration	**MonotonyEnum**
noMonotony	This indicates that the related curve needs not to be monotony. **Tags:** atp.EnumerationLiteralIndex=3
strictlyDecreasing	This indicates that the related curve needs to be strictly monotony decreasing. **Tags:** atp.EnumerationLiteralIndex=4
strictlyIncreasing	This indicates that the related curve needs to be strictly monotony increasing. **Tags:** atp.EnumerationLiteralIndex=5
strictMonotonous	This indicates that the values shall be strict monotonously decreasing or increasing, depending on the trend set by the first values of the series. **Tags:** atp.EnumerationLiteralIndex=6

Table B.95: MonotonyEnum

Class	**NonqueuedReceiverComSpec**			
Note	Communication attributes specific to non-queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aliveTimeout	TimeValue	0..1	attr	Specify the amount of time (in seconds) after which the software component (via the RTE) needs to be notified if the corresponding data item have not been received according to the specified timing description. If the aliveTimeout attribute is 0 no timeout monitoring shall be performed. This Attribute is only used by the AUTOSAR Classic Platform.
enableUpdate	Boolean	0..1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding Variable DataPrototype has been updated. This Attribute is only used by the AUTOSAR Classic Platform.
filter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement. **Tags:** atp.Splitkey=filter
handleData Status	Boolean	0..1	attr	If this attribute is set to true, then the Rte_IStatus API shall exist. If the attribute does not exist or is set to false, then the Rte_IStatus API may still exist in response to the existence of further conditions. This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	**NonqueuedReceiverComSpec**			
handleNever Received	Boolean	0..1	attr	This attribute specifies whether for the corresponding VariableDataPrototype the "never received" flag is available. If yes, the RTE is supposed to assume that initially the VariableDataPrototype has not been received before. After the first reception of the corresponding VariableDataPrototype the flag is cleared. • If the value of this attribute is set to "true" the flag is required. • If set to "false", the RTE shall not support the "never received" functionality for the corresponding Variable DataPrototype. This Attribute is only used by the AUTOSAR Classic Platform.
handleTimeout Type	HandleTimeoutEnum	0..1	attr	This attribute controls the behavior with respect to the handling of timeouts. This Attribute is only used by the AUTOSAR Classic Platform.
initValue	ValueSpecification	0..1	aggr	Initial value to be used in case the sending component is not yet initialized. If the sender also specifies an initial value, then the receiver's value will be used. **Tags:** atp.Splitkey=initValue This Attribute is only used by the AUTOSAR Classic Platform.
returnNoNew DataEnabled	Boolean	0..1	attr	This attribute defines whether the RTE API functions related to the RPortPrototype shall return No New Data Error if no new data is received from COM. This Attribute is only used by the AUTOSAR Classic Platform.
timeout Substitution Value	ValueSpecification	0..1	aggr	This attribute represents the substitution value applicable in the case of a timeout. **Tags:** atp.Splitkey=timeoutSubstitutionValue This Attribute is only used by the AUTOSAR Classic Platform.
transportError CountEnabled	Boolean	0..1	attr	This attribute defines whether the RTE API functions related to the RPortPrototype shall return the number of transport errors (i.e. COM, SecOC errors) that happened since the last call of the respective API. This Attribute is only used by the AUTOSAR Classic Platform.
valueErrorCount Enabled	Boolean	0..1	attr	This attribute defines whether the RTE API functions related to the RPortPrototype shall return the number of value errors (i.e. out of range, invalid value) that happened since the last call of the respective API. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.96: NonqueuedReceiverComSpec

Class	**NonqueuedSenderComSpec**			
Note	Communication attributes for non-queued sender/receiver communication (sender side)			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**NonqueuedSenderComSpec**			
dataFilter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement. **Tags:** atp.Splitkey=dataFilter
initValue	ValueSpecification	0..1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already. **Tags:** atp.Splitkey=initValue

Table B.97: NonqueuedSenderComSpec

Class	**NvBlockNeeds**			
Note	Specifies the abstract needs on the configuration of a single NVRAM Block.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Aggregated by	BswServiceDependency.serviceNeeds, NvBlockDescriptor.nvBlockNeeds, SwcService Dependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
calcRamBlock Crc	Boolean	0..1	attr	Defines if CRC (re)calculation for the permanent RAM Block is required.
checkStatic BlockId	Boolean	0..1	attr	Defines if the Static Block Id check shall be enabled.
cyclicWriting Period	TimeValue	0..1	attr	This represents the period for cyclic writing of NvData to store the associated RAM Block.
nDataSets	PositiveInteger	0..1	attr	Number of data sets to be provided by the NVRAM manager for this block. This is the total number of ROM Blocks and RAM Blocks.
nRomBlocks	PositiveInteger	0..1	attr	Number of ROM Blocks to be provided by the NVRAM manager for this block. Please note that these multiple ROM Blocks are given in a contiguous area.
ramBlockStatus Control	RamBlockStatusControl Enum	0..1	attr	This attribute defines how the management of the RAM Block status is controlled.
readonly	Boolean	0..1	attr	true: data of this NVRAM Block are write protected for normal operation (but protection can be disabled) false: no restriction
reliability	NvBlockNeeds ReliabilityEnum	0..1	attr	Reliability against data loss on the non-volatile medium.
resistantTo ChangedSw	Boolean	0..1	attr	Defines whether an NVRAM Block shall be treated resistant to configuration changes (true) or not (false). For details how to handle initialization in the latter case, please refer to the NVRAM specification.
restoreAtStart	Boolean	0..1	attr	Defines whether the associated RAM Block shall be implicitly restored during startup by the basic software.
selectBlockFor FirstInitAll	Boolean	0..1	attr	If this attribute is set to true the NvM shall process this block in the NvM_FirstInitAll() function.
storeAt Shutdown	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored during shutdown by the basic software.
storeCyclic	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored periodically by the basic software.
store Emergency	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored in case of ECU failure (e.g. loss of power) by the basic software. If the attribute store Emergency is set to true the associated RAM Block shall be configured to have immediate priority.

▽

△

Class	NvBlockNeeds			
storeImmediate	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored immediately during or after execution of the according SW-C RunnableEntity by the basic software.
storeOnChange	Boolean	0..1	attr	This attribute defines whether the associated RAM Block shall be stored immediately if the written value is different to the value stored in the associated RAM Block(s) during or after execution of the according SW-C RunnableEntity.
useAuto ValidationAt ShutDown	Boolean	0..1	attr	If set to true the RAM Block shall be auto validated during shutdown phase.
useCRCComp Mechanism	Boolean	0..1	attr	If set to true the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job in order to skip unnecessary NVRAM writings.
writeOnlyOnce	Boolean	0..1	attr	Defines write protection after first write: true: This block is prevented from being changed/erased or being replaced with the default ROM data after first initialization by the software-component. false: No such restriction.
writeVerification	Boolean	0..1	attr	Defines if Write Verification shall be enabled for this NVRAM Block.
writing Frequency	PositiveInteger	0..1	attr	Provides the amount of updates to this block from the application point of view. It has to be provided in "number of write access per year".
writingPriority	NvBlockNeedsWriting PriorityEnum	0..1	attr	Requires the priority of writing this block in case of concurrent requests to write other blocks.

Table B.98: NvBlockNeeds

Class	NvDataInterface			
Note	A non volatile data interface declares a number of `VariableDataPrototype`s to be exchanged between non volatile block components and atomic software components. **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**
nvData	VariableDataPrototype	*	aggr	The VariableDataPrototype of this nv data interface. **Tags:** atp.Splitkey=nvData, nvData.shortName

Table B.99: NvDataInterface

Class	OperationInvokedEvent			
Note	This event is raised when the `ClientServerOperation` referenced in `OperationInvokedEvent.operation` shall be invoked.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	OperationInvokedEvent			
operation	ClientServerOperation	0..1	iref	This represents the ClientServerOperation which shall be invoked. **Tags:** atp.Splitkey=operation, operation.contextPPort, operation.targetProvidedOperation **InstanceRef implemented by:** POperationInAtomicSwcInstanceRef

Table B.100: OperationInvokedEvent

Class	OsTaskExecutionEvent			
Note	This `RTEEvent` is supposed to execute `RunnableEntity`s which have to react on the execution of specific OsTasks. Therefore, this event is unconditionally raised whenever the OsTask on which it is mapped is executed. The main use case for this event is scheduling of `RunnableEntity`s of Complex Drivers which have to react on task executions.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.101: OsTaskExecutionEvent

Class	PPortPrototype			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
providedInterface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType **Tags:** atp.Splitkey=providedInterface

Table B.102: PPortPrototype

Class	PRPortPrototype			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
providedRequiredInterface	PortInterface	0..1	tref	This represents the `PortInterface` used to type the `PRPortPrototype`. **Stereotypes:** isOfType **Tags:** atp.Splitkey=providedRequiredInterface

Table B.103: PRPortPrototype

Class	ParameterAccess			
Note	The presence of a ParameterAccess implies that a RunnableEntity needs access to a ParameterDataPrototype.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, RunnableEntity.parameterAccess			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessed Parameter	AutosarParameterRef	0..1	aggr	Reference to the accessed calibration parameter. **Tags:** atp.Splitkey=accessedParameter
swDataDef Props	SwDataDefProps	0..1	aggr	This allows denote instance and access specific properties, mainly input values and common axis. **Tags:** atp.Splitkey=swDataDefProps

Table B.104: ParameterAccess

Class	ParameterDataPrototype			
Note	A ParameterDataPrototype represents a formalized generic piece of information that is typically immutable by the application software layer, but mutable by measurement and calibration tools. ParameterDataPrototype is used in various contexts and the specific context gives the otherwise generic ParameterDataPrototype a dedicated semantics.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, BswInternalBehavior.perInstanceParameter, *InternalBehavior*.constantMemory, NvBlockDescriptor.romBlock, ParameterInterface.parameter, SwcInternalBehavior.perInstanceParameter, SwcInternalBehavior.sharedParameter			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the ParameterDataPrototype **Tags:** atp.Splitkey=initValue

Table B.105: ParameterDataPrototype

Class	ParameterInterface			
Note	A parameter interface declares a number of parameter and characteristic values to be exchanged between parameter components and software components. **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**
parameter	ParameterDataPrototype	*	aggr	The ParameterDataPrototype of this ParameterInterface. **Tags:** atp.Splitkey=parameter, parameter.shortName

Table B.106: ParameterInterface

Class	ParameterRequireComSpec			
Note	"Communication" specification that applies to parameters on the required side of a connection.			
Base	*ARObject*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	ParameterRequireComSpec			
initValue	ValueSpecification	0..1	aggr	The initial value applicable for the corresponding ParameterDataPrototype. **Tags:** atp.Splitkey=initValue
parameter	ParameterDataPrototype	0..1	ref	The ParameterDataPrototype to which the Parameter RequireComSpec applies. **Stereotypes:** atpIdentityContributor

Table B.107: ParameterRequireComSpec

Class	PhysConstrs			
Note	This meta-class represents the ability to express physical constraints. Therefore it has (in opposite to InternalConstrs) a reference to a Unit.			
Base	*ARObject*			
Aggregated by	DataConstrRule.physConstrs			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. **Tags:** xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in curves and maps. **Tags:** xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This specifies the monotony constraints on the data object. Note that this applies only to curves and maps. **Tags:** xml.sequenceOffset=70
unit	Unit	0..1	ref	This is the unit to which the physical constraints relate to. In particular, it is the physical unit of the specified limits. **Tags:** atp.Splitkey=unit xml.sequenceOffset=80
upperLimit	Limit	0..1	attr	This specifies the upper limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table B.108: PhysConstrs

Class	PhysicalDimension			
Note	This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension. Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm). Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2. **Tags:** atp.recommendedPackage=PhysicalDimensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". **Tags:** xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". **Tags:** xml.sequenceOffset=20
luminous IntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". **Tags:** xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". **Tags:** xml.sequenceOffset=30
molarAmount Exp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". **Tags:** xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". **Tags:** xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". **Tags:** xml.sequenceOffset=40

Table B.109: PhysicalDimension

Class	PortElementToCommunicationResourceMapping			
Note	This meta class maps a communication resource to CP Software Clusters. In this case the kind of Port Prototype specified whether the Software Cluster has to provide or to require the resource. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CpSoftwareClusterMappingSet.portElementToComResourceMapping, SystemMapping.portElementToComResourceMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clientServer Operation	ClientServerOperation	0..1	iref	ClientServerOperation instance qualifying the communication resource **Tags:** atp.Splitkey=clientServerOperation, clientServer Operation.contextComposition, clientServer Operation.contextComponent, clientServer Operation.contextPort, clientServerOperation.target Operation **InstanceRef implemented by:** OperationInSystem InstanceRef

▽

△

Class	**PortElementToCommunicationResourceMapping**			
communication Resource	CpSoftwareCluster Communication Resource	0..1	ref	Communication resource for which the mapping applies. **Tags:** atp.Splitkey=communicationResource
mode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	iref	ModeDeclarationGroupPrototype instance qualifying the communication resource **Tags:** atp.Splitkey=modeDeclarationGroupPrototype, modeDeclarationGroupPrototype.contextComposition, modeDeclarationGroupPrototype.contextComponent, modeDeclarationGroupPrototype.contextPort, mode DeclarationGroupPrototype.targetModeDeclarationGroup Prototype **InstanceRef implemented by:** ModeDeclarationGroup PrototypeInSystemInstanceRef
parameterData Prototype	ParameterData Prototype	0..1	iref	ParameterDataPrototype instance qualifying the communication resource. **Tags:** atp.Splitkey=parameterDataPrototype, parameter DataPrototype.contextComposition, parameterData Prototype.contextComponent, parameterData Prototype.contextPort, parameterDataPrototype.target ParameterDataPrototype **InstanceRef implemented by:** ParameterDataPrototype InSystemInstanceRef
trigger	Trigger	0..1	iref	Trigger instance qualifying the communication resource. **Tags:** atp.Splitkey=trigger, trigger.contextComposition, trigger.contextComponent, trigger.contextPort, trigger.targetTrigger **InstanceRef implemented by:** TriggerInSystemInstance Ref
variableData Prototype	VariableDataPrototype	0..1	iref	VariableDataPrototype instance qualifying the communication resource **Tags:** atp.Splitkey=variableDataPrototype, variableData Prototype.contextComponent, variableData Prototype.contextComposition, variableData Prototype.contextPort, variableDataPrototype.targetData Prototype **InstanceRef implemented by:** VariableDataPrototypeIn SystemInstanceRef

Table B.110: PortElementToCommunicationResourceMapping

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***

▽

△

Class	***PortInterface*** (abstract)			
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 B.111: PortInterface

Class	***PortInterfaceMapping*** (abstract)			
Note	Specifies one `PortInterfaceMapping` to support the connection of Ports typed by two different `PortInterface`s with PortInterface elements having unequal names and/or unequal semantic (resolution or range).			
Base	*ARObject*, *AtpBlueprint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ClientServerInterfaceMapping, ModeInterfaceMapping, TriggerInterfaceMapping, VariableAndParameter InterfaceMapping			
Aggregated by	PortInterfaceMappingSet.portInterfaceMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.112: PortInterfaceMapping

Class	***PortPrototype*** (abstract)			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this `PortPrototype` with respect to client/server communication. **Tags:** atp.Splitkey=clientServerAnnotation
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port. **Tags:** atp.Splitkey=delegatedPortAnnotation

▽

△

Class	***PortPrototype*** (abstract)			
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port. **Tags:** atp.Splitkey=ioHwAbstractionServerAnnotation
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port. **Tags:** atp.Splitkey=modePortAnnotation
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port. **Tags:** atp.Splitkey=nvDataPortAnnotation
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port. **Tags:** atp.Splitkey=parameterPortAnnotation
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication. **Tags:** atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port. **Tags:** atp.Splitkey=triggerPortAnnotation

Table B.113: PortPrototype

Class	**QueuedReceiverComSpec**			
Note	Communication attributes specific to queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
queueLength	PositiveInteger	0..1	attr	Length of queue for received events. This Attribute is only used by the AUTOSAR Classic Platform.

Table B.114: QueuedReceiverComSpec

Class	**RModeInAtomicSwcInstanceRef**			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	*RTEEvent*.disabledMode, SwcModeSwitchEvent.mode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	AtomicSwComponent Type	0..1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** xml.sequenceOffset=30
contextPort	AbstractRequiredPort Prototype	0..1	ref	**Tags:** xml.sequenceOffset=20
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** xml.sequenceOffset=40

Table B.115: RModeInAtomicSwcInstanceRef

Class	**RPortPrototype**			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *Multilanguage Referrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
mayBe Unconnected	Boolean	0..1	attr	If set to true, this attribute indicates that the enclosing `RPortPrototype` may be left unconnected and that this aspect has explicitly been considered in the software-component's design. This Attribute is only used by the AUTOSAR Classic Platform.
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType **Tags:** atp.Splitkey=requiredInterface

Table B.116: RPortPrototype

Class	***RTEEvent*** (abstract)			
Note	Abstract base class for all RTE-related events			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Subclasses	AsynchronousServerCallReturnsEvent, BackgroundEvent, DataReceiveErrorEvent, DataReceivedEvent, DataSendCompletedEvent, DataWriteCompletedEvent, ExternalTriggerOccurredEvent, InitEvent, InternalTriggerOccurredEvent, ModeSwitchedAckEvent, OperationInvokedEvent, OsTaskExecutionEvent, SwcModeManagerErrorEvent, SwcModeSwitchEvent, TimingEvent, TransformerHardErrorEvent			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
disabledMode	ModeDeclaration	*	iref	Reference to the Modes that disable the Event. **Tags:** atp.Splitkey=disabledMode, disabledMode.contextPort, disabledMode.contextModeDeclarationGroupPrototype, disabledMode.targetModeDeclaration **InstanceRef implemented by:** RModeInAtomicSwcInstanceRef
startOnEvent	RunnableEntity	0..1	ref	The referenced `RunnableEntity` starts when the corresponding `RTEEvent` is raised. **Tags:** atp.Splitkey=startOnEvent

Table B.117: RTEEvent

Class	**RTriggerInAtomicSwcInstanceRef**			
Note				
Base	*ARObject*, *AtpInstanceRef*, *TriggerInAtomicSwcInstanceRef*			
Aggregated by	ExternalTriggerOccurredEvent.trigger, TransformerHardErrorEvent.requiredTrigger			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contextRPort	AbstractRequiredPort Prototype	0..1	ref	**Tags:** xml.sequenceOffset=20
targetTrigger	Trigger	0..1	ref	**Tags:** xml.sequenceOffset=30

Table B.118: RTriggerInAtomicSwcInstanceRef

Class	**Referrable** (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, TpConnectionIdent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortNameFragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** atp.Splitkey=shortNameFragment xml.sequenceOffset=-90

Table B.119: Referrable

Class	**RtePluginProps**			
Note	The properties of a communication graph with respect to the utilization of RTE Implementation Plug-in. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*			
Aggregated by	FlatInstanceDescriptor.rtePluginProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
associatedCrossSwClusterComRtePlugin	EcucContainerValue	0..1	ref	This associates a communication graph to a specific RTE Implementation Plug-in handling cross Software Cluster communication. **Tags:** atp.Splitkey=associatedCrossSwClusterComRtePlugin
associatedRtePlugin	EcucContainerValue	0..1	ref	This associates a communication graph to a specific RTE Implementation Plug-in handling local Software Cluster communication or communication in a non-cluster ECU. **Tags:** atp.Splitkey=associatedRtePlugin

Table B.120: RtePluginProps

Class	**RunnableEntity**			
Note	A `RunnableEntity` represents the smallest code-fragment that is provided by an `AtomicSwComponentType` and are executed under control of the RTE. `RunnableEntity`s are for instance set up to respond to data reception or operation invocation on a server.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.runnable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
argument (ordered)	RunnableEntityArgument	*	aggr	This represents the formal definition of an argument to a RunnableEntity.

▽

△

Class	**RunnableEntity**			
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	aggr	The server call result point admits a runnable to fetch the result of an asynchronous server call. The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=asynchronousServerCallResultPoint, asynchronousServerCallResultPoint.shortName, asynchronousServerCallResultPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.
canBeInvoked Concurrently	Boolean	0..1	attr	If the value of this attribute is set to "true" the enclosing `RunnableEntity` can be invoked concurrently (even for one instance of the corresponding `AtomicSwComponentType`). This implies that it is the responsibility of the implementation of the `RunnableEntity` to take care of this form of concurrency.
dataRead Access	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataReadAccess, dataReadAccess.shortName, dataReadAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointBy Argument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataReceivePointByArgument, dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
dataReceive PointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointBy Value is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataReceivePointByValue, dataReceivePoint ByValue.shortName, dataReceivePointByValue.variation Point.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataSendPoint, dataSendPoint.shortName, dataSendPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataWrite Access	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=dataWriteAccess, dataWriteAccess.short Name, dataWriteAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
external TriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=externalTriggeringPoint, externalTriggering Point.ident.shortName, externalTriggeringPoint.variation Point.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint, internalTriggering Point.shortName, internalTriggeringPoint.variation Point.shortLabel vh.latestBindingTime=preCompileTime
modeAccess Point	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=modeAccessPoint, modeAccess Point.ident.shortName, modeAccessPoint.variation Point.shortLabel vh.latestBindingTime=preCompileTime
modeSwitch Point	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=modeSwitchPoint, modeSwitchPoint.short Name, modeSwitchPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
parameter Access	ParameterAccess	*	aggr	The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype. The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of Parameter Access (points) in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=parameterAccess, parameterAccess.short Name, parameterAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	**RunnableEntity**			
readLocal Variable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=readLocalVariable, readLocalVariable.short Name, readLocalVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=serverCallPoint, serverCallPoint.shortName, serverCallPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime This Attribute is only used by the AUTOSAR Classic Platform.
symbol	CIdentifier	0..1	attr	The symbol describing this `RunnableEntity`'s entry point. This is considered the API of the `RunnableEntity` and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The `WaitPoint` associated with the `RunnableEntity`. **Tags:** atp.Splitkey=waitPoint, waitPoint.shortName
writtenLocal Variable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=writtenLocalVariable, writtenLocal Variable.shortName, writtenLocalVariable.variation Point.shortLabel vh.latestBindingTime=preCompileTime

Table B.121: RunnableEntity

Enumeration	**SendIndicationEnum**
Note	This meta-class provides a way to specify in which way redundancy shall be applied on collection level. This Enumeration is only used by the AUTOSAR Classic Platform.
Aggregated by	DataComProps.sendIndication
Literal	***Description***

▽

△

Enumeration	**SendIndicationEnum**
anySendOperation	This value represents the requirement that any send operation of the Software Cluster is indicated. **Tags:** atp.EnumerationLiteralIndex=2
none	This value represents the requirement that send operations of the Software Cluster are not indicated. **Tags:** atp.EnumerationLiteralIndex=1

Table B.122: SendIndicationEnum

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 B.123: SenderReceiverInterface

Class	***ServerCallPoint*** (abstract)			
Note	If a RunnableEntity owns a `ServerCallPoint` it is entitled to invoke a particular `ClientServerOperation` of a specific `RPortPrototype` of the corresponding `AtomicSwComponentType`. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AsynchronousServerCallPoint, SynchronousServerCallPoint			
Aggregated by	*AtpClassifier*.atpFeature, RunnableEntity.serverCallPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operation	ClientServerOperation	0..1	iref	The operation that is called by this runnable. **Tags:** atp.Splitkey=operation, operation.contextRPort, operation.targetRequiredOperation **InstanceRef implemented by:** ROperationInAtomicSwcInstanceRef
timeout	TimeValue	0..1	attr	Time in seconds before the server call times out and returns with an error message. It depends on the call type (synchronous or asynchronous) how this is reported.

Table B.124: ServerCallPoint

Class	ServerComSpec			
Note	Communication attributes for a server port (PPortPrototype and ClientServerInterface).			
Base	*ARObject*, *PPortComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
operation	ClientServerOperation	0..1	ref	Operation these communication attributes apply to. **Stereotypes:** atpIdentityContributor
queueLength	PositiveInteger	0..1	attr	Length of call queue on the server side. The queue is implemented by the RTE. The value shall be greater or equal to 1. Setting the value of queueLength to 1 implies that incoming requests are rejected while another request that arrived earlier is being processed.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation. **Tags:** atp.Splitkey=transformationComSpecProps

Table B.125: ServerComSpec

Class	ServiceSwComponentType			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Collectable Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.126: ServiceSwComponentType

Class	SubElementMapping			
Note	This meta-class allows for the definition of mappings of elements of a composite data type.			
Base	*ARObject*			
Aggregated by	DataPrototypeMapping.subElementMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
firstElement	SubElementRef	0..1	aggr	This represents the first element referenced in the scope of the mapping. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=firstElement, firstElement.variation Point.shortLabel vh.latestBindingTime=preCompileTime
secondElement	SubElementRef	0..1	aggr	This represents the second element referenced in the scope of the mapping. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=secondElement, secondElement.variation Point.shortLabel vh.latestBindingTime=preCompileTime

▽

△

Class	SubElementMapping			
textTable Mapping	TextTableMapping	*	aggr	This allows for the text-table translation of individual elements of a composite data type. **Tags:** atp.Splitkey=textTableMapping

Table B.127: SubElementMapping

Class	SwBaseType			
Note	This meta-class represents a base type used within ECU software. **Tags:** atp.recommendedPackage=BaseTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *BaseType*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table B.128: SwBaseType

Enumeration	SwCalibrationAccessEnum
Note	Determines the access rights to a data object w.r.t. measurement and calibration.
Aggregated by	ModeDeclarationGroupPrototype.swCalibrationAccess, SwCalprmAxis.swCalibrationAccess, SwDataDefProps.swCalibrationAccess
Literal	Description
notAccessible	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file. **Tags:** atp.EnumerationLiteralIndex=0
readOnly	The element will only appear as read-only in an ASAP file. **Tags:** atp.EnumerationLiteralIndex=1
readWrite	The element will appear in the ASAP file with both read and write access. **Tags:** atp.EnumerationLiteralIndex=2

Table B.129: SwCalibrationAccessEnum

Class	SwComponentPrototype			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.component			
Attribute	Type	Mult.	Kind	Note
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table B.130: SwComponentPrototype

Class	«atpVariation» **SwDataDefProps**
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swAddrMethod, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalid Value • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime
Base	*ARObject*
Aggregated by	*AutosarDataType*.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, Cpp ImplementationDataTypeElement.swDataDefProps, *DataPrototype*.swDataDefProps, DataPrototype TransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, Diagnostic EnvDataElementCondition.swDataDefProps, DiagnosticExtendedDataRecordElement.swDataDefProps, DiagnosticSovdPrimitiveContentElement.swDataDefProps, DltArgumentProps.networkRep, FlatInstance Descriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDef Props.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, *ReceiverComSpec*.network Representation, SecurityEventContextDataElement.networkRepresentation, *SenderComSpec*.network Representation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTarget Props.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, System Signal.physicalProps

Attribute	**Type**	**Mult.**	**Kind**	**Note**
additionalNative TypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:** xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** atp.Splitkey=annotation xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:** atp.Splitkey=baseType xml.sequenceOffset=50



△

Class	«atpVariation» **SwDataDefProps**			
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:** atp.Splitkey=compuMethod xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:** atp.Splitkey=dataConstr xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:** xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** atp.Splitkey=implementationDataType xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** atp.Splitkey=invalidValue xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** atp.Splitkey=swAddrMethod xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod. **Tags:** xml.sequenceOffset=33

▽

△

Class	«atpVariation» **SwDataDefProps**			
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** atp.Splitkey=swBitRepresentation xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** atp.Splitkey=swCalprmAxisSet xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** atp.Splitkey=swComparisonVariable xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** atp.Splitkey=swDataDependency xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** atp.Splitkey=swHostVariable xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250

▽

△

Class	«atpVariation» **SwDataDefProps**			
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** atp.Splitkey=swPointerTargetProps xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** atp.Splitkey=swRecordLayout xml.sequenceOffset=290
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** atp.Splitkey=swRefreshTiming xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** atp.Splitkey=swTextProps xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** atp.Splitkey=unit xml.sequenceOffset=350

▽

△

Class	«atpVariation» **SwDataDefProps**			
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** atp.Splitkey=valueAxisDataType xml.sequenceOffset=355

Table B.131: SwDataDefProps

Enumeration	SwImplPolicyEnum
Note	Specifies the implementation strategy with respect to consistency mechanisms of variables.
Aggregated by	BswInternalTriggeringPoint.swImplPolicy, InternalTriggeringPoint.swImplPolicy, SwDataDefProps.swImplPolicy, Trigger.swImplPolicy
Literal	Description
const	forced implementation such that the running software within the ECU shall not modify it. For example implemented with the "const" modifier in C. This can be applied for parameters (not for those in NVRAM) as well as argument data prototypes. **Tags:** atp.EnumerationLiteralIndex=0
fixed	This data element is fixed. In particular this indicates, that it might also be implemented e.g. as in place data, (#DEFINE). **Tags:** atp.EnumerationLiteralIndex=1
measurementPoint	The data element is created for measurement purposes only. The data element is never read directly within the ECU software. In contrast to a "standard" data element in an unconnected provide port is, this unconnection is guaranteed for measurementPoint data elements. **Tags:** atp.EnumerationLiteralIndex=2
queued	The content of the data element is queued and the data element has 'event' semantics, i.e. data elements are stored in a queue and all data elements are processed in 'first in first out' order. The queuing is intended to be implemented by RTE Generator. This value is not applicable for parameters. **Tags:** atp.EnumerationLiteralIndex=3
standard	This is applicable for all kinds of data elements. For variable data prototypes the 'last is best' semantics applies. For parameter there is no specific implementation directive. **Tags:** atp.EnumerationLiteralIndex=4

Table B.132: SwImplPolicyEnum

Class	SwRecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:** atp.recommendedPackage=SwRecordLayouts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note

▽

△

Class	SwRecordLayout			
swRecord LayoutGroup	SwRecordLayoutGroup	0..1	aggr	This is the top level record layout group. **Tags:** atp.Splitkey=swRecordLayoutGroup xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table B.133: SwRecordLayout

Class	SwcModeManagerErrorEvent			
Note	This event is raised when an error occurred during the handling of the referenced ModeDeclarationGroup Prototype.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
modeGroup	ModeDeclarationGroup Prototype	0..1	iref	This represents the ModeDeclarationGroupPrototype for which this SwcModeManagerErrorEvent is raised in case of an error. **Tags:** atp.Splitkey=modeGroup, modeGroup.context PPort, modeGroup.targetModeGroup **InstanceRef implemented by:** PModeGroupInAtomic SwcInstanceRef

Table B.134: SwcModeManagerErrorEvent

Class	SwcModeSwitchEvent			
Note	This event is raised when the specified mode change occurs.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activation	ModeActivationKind	0..1	attr	Specifies if the event is raised on entering or exiting a specific mode or is raised on the transition between two modes.
mode (ordered)	ModeDeclaration	0..2	iref	The referenced mode or the transition between two modes raises this SwcModeSwitchEvent. **InstanceRef implemented by:** RModeInAtomicSwc InstanceRef

Table B.135: SwcModeSwitchEvent

Class	SynchronousServerCallPoint
Note	This means that the RunnableEntity is supposed to perform a blocking wait for a response from the server. This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServerCallPoint*
Aggregated by	*AtpClassifier*.atpFeature, RunnableEntity.serverCallPoint

▽

△

Class	**SynchronousServerCallPoint**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
calledFrom WithinExclusive Area	ExclusiveAreaNesting Order	0..1	ref	This indicates that the call point is located at the deepest level inside one or more ExclusiveAreas that are nested in the given order. **Tags:** atp.Splitkey=calledFromWithinExclusiveArea

Table B.136: SynchronousServerCallPoint

Class	**System**			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientId DefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System. **Tags:** atp.Splitkey=clientIdDefinitionSet This Attribute is only used by the AUTOSAR Classic Platform.
containerIPdu HeaderByte Order	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus. This Attribute is only used by the AUTOSAR Classic Platform.
ecuExtract Version	RevisionLabelString	0..1	attr	Version number of the Ecu Extract. This Attribute is only used by the AUTOSAR Classic Platform.
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=fibexElement, fibexElement.fibexElement, fibexElement.variationPoint.shortLabel vh.latestBindingTime=postBuild
interpolation Routine MappingSet	InterpolationRoutine MappingSet	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System. **Tags:** atp.Splitkey=interpolationRoutineMappingSet This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	System			
j1939Shared AddressCluster	J1939SharedAddress Cluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=j1939SharedAddressCluster, j1939Shared AddressCluster.shortName, j1939SharedAddress Cluster.variationPoint.shortLabel vh.latestBindingTime=postBuild This Attribute is only used by the AUTOSAR Classic Platform.
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of System Mapping can be provided by several parties using different names for the SystemMapping. This element is not required when the System description is used for a network-only use-case. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=mapping, mapping.shortName, mapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=rootSoftwareComposition, rootSoftware Composition.shortName, rootSoftware Composition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime This Attribute is only used by the AUTOSAR Classic Platform.
swCluster	CpSoftwareCluster	*	ref	CP Software Clusters of this System **Stereotypes:** atpVariation **Tags:** atp.Splitkey=swCluster, swCluster.cpSoftwareCluster, sw Cluster.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime This Attribute is only used by the AUTOSAR Classic Platform.
systemCom SpecDefinition	SystemComSpec DefinitionSet	*	ref	Reference to the set of ComSpec definitions that are used for inter-ECU communication in the System. **Tags:** atp.Splitkey=systemComSpecDefinition

▽

△

Class	System			
system Documentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=systemDocumentation, system Documentation.shortName, system Documentation.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10 This Attribute is only used by the AUTOSAR Classic Platform.
systemVersion	RevisionLabelString	0..1	attr	Version number of the System Description.

Table B.137: System

Class	TimingEvent			
Note	This event is used to start `RunnableEntity`s that shall be executed periodically.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
offset	TimeValue	0..1	attr	The value makes an assumption about the time offset of the first activation of the `RunnableEntity` triggered by the mapped `TimingEvent` relative to the periodic activation of the time base of this `TimingEvent`. Unit: second.
period	TimeValue	0..1	attr	Period of timing event in seconds. The value of this attribute shall be greater than zero.

Table B.138: TimingEvent

Class	TransformerHardErrorEvent			
Note	This event is raised when data are received which should trigger a Client/Server operation or an external Trigger but during transformation of the data a hard transformer error occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SwcInternalBehavior.event			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
operation	ClientServerOperation	0..1	iref	This represents the ClientServerOperation for which the transformer can raise this TransformerHardErrorEvent. **Tags:** atp.Splitkey=operation, operation.contextPPort, operation.targetProvidedOperation **InstanceRef implemented by:** POperationInAtomicSwc InstanceRef
requiredTrigger	Trigger	0..1	iref	This represents the Trigger for which the transformer can raise this TransformerHardErrorEvent. **Tags:** atp.Splitkey=requiredTrigger, required Trigger.contextRPort, requiredTrigger.targetTrigger **InstanceRef implemented by:** RTriggerInAtomicSwc InstanceRef

Table B.139: TransformerHardErrorEvent

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.requiredTrigger, ServiceInterface.trigger, TriggerInterface.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 B.140: 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 B.141: TriggerInterface

Class	Unit			
Note	This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (factorSiToUnit) and the offset (offsetSiTo Unit) are applied as follows: x [{unit}] := y * {siUnit} * factorSiToUnit [{unit}/{siUnit}] + offsetSiToUnit [{unit}] For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied. y [{siUnit}] := (x*{unit} - offsetSiToUnit [{unit}]) / factorSiToUnit [{unit}/{siUnit}] **Tags:** atp.recommendedPackage=Units			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	Unit			
displayName	SingleLanguageUnit Names	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. **Tags:** atp.Splitkey=displayName xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units. **Tags:** xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. **Tags:** xml.sequenceOffset=40
physical Dimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. **Tags:** atp.Splitkey=physicalDimension xml.sequenceOffset=50

Table B.142: Unit

Class	VariableAccess			
Note	The presence of a `VariableAccess` implies that a `RunnableEntity` needs access to a `VariableDataPrototype`. The kind of access is specified by the role in which the class is used.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *ReceiverComSpec*.replaceWith, RunnableEntity.dataReadAccess, RunnableEntity.dataReceivePointByArgument, RunnableEntity.dataReceivePointByValue, RunnableEntity.dataSendPoint, RunnableEntity.dataWriteAccess, RunnableEntity.readLocalVariable, RunnableEntity.writtenLocalVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessed Variable	AutosarVariableRef	0..1	aggr	This denotes the accessed variable. **Tags:** atp.Splitkey=accessedVariable
scope	VariableAccessScope Enum	0..1	attr	This attribute allows for constraining the scope of the corresponding communication. For example, it possible to express whether the communication is intended to cross the boundary of an ECU or whether it is intended not to cross the boundary of a single partition.

Table B.143: VariableAccess

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*

▽

△

Class	**VariableDataPrototype**			
Aggregated by	ApplicationInterface.indication, *AtpClassifier*.atpFeature, BswInternalBehavior.arTypedPerInstance Memory, BswModuleDescription.providedData, BswModuleDescription.requiredData, BulkNvData Descriptor.bulkNvBlock, DiagnosticSovdAccessArgument.contentObject, *InternalBehavior*.staticMemory, NvBlockDescriptor.ramBlock, NvDataInterface.nvData, SenderReceiverInterface.dataElement, Service Interface.event, SwcInternalBehavior.arTypedPerInstanceMemory, SwcInternalBehavior.explicitInter RunnableVariable, SwcInternalBehavior.implicitInterRunnableVariable			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype **Tags:** atp.Splitkey=initValue

Table B.144: VariableDataPrototype

C Referenced ECUC Configuration Parameters

C.1 EcuC

C.1.1 EcucPartition

[ECUC_EcuC_00005] Definition of EcucParamConfContainerDef EcucPartition

Container Name	EcucPartition
Parent Container	EcucPartitionCollection
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Definition of one Partition on this ECU. One Partition will be implemented using one Os-Application.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucPartitionId	1	[ECUC_EcuC_00085]
EcucEcuPartitionRef	0..1	[ECUC_EcuC_00083]
EcucPartitionBswModuleDistinguishedPartition	0..*	[ECUC_EcuC_00068]
EcucPartitionCoreRef	1	[ECUC_EcuC_00086]
EcucPartitionSoftwareComponentInstanceRef	0..*	[ECUC_EcuC_00036]

No Included Containers

[ECUC_EcuC_00085] Definition of EcucIntegerParamDef EcucPartitionId

Parameter Name	EcucPartitionId	
Parent Container	EcucPartition	
Upstream mapping defined	–	
Description	ID of the partition.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_EcuC_00083] Definition of EcucForeignReferenceDef EcucEcuPartitionRef

Parameter Name	EcucEcuPartitionRef
Parent Container	EcucPartition
Upstream mapping defined	–

▽

△

Description	Reference to the EcuPartition to define the link to the partition described in the System description. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Foreign reference to ECU-PARTITION
Pre-Compile only	true

[ECUC_EcuC_00068] Definition of EcucForeignReferenceDef EcucPartitionBswModuleDistinguishedPartition

Parameter Name	EcucPartitionBswModuleDistinguishedPartition
Parent Container	EcucPartition
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate]
Description	This maps the abstract partition of the Bsw Module to a concrete Partition existing in the ECU.
Multiplicity	0..*
Type	Foreign reference to BSW-DISTINGUISHED-PARTITION
Pre-Compile only	true

[ECUC_EcuC_00086] Definition of EcucReferenceDef EcucPartitionCoreRef

Parameter Name	EcucPartitionCoreRef
Parent Container	EcucPartition
Upstream mapping defined	–
Description	Reference to the core definition. This reference is used to describe to which core the EcucPartition is bound.
Multiplicity	1
Type	Reference to EcucCoreDefinition
Pre-Compile only	true

[ECUC_EcuC_00036] Definition of EcucInstanceReferenceDef EcucPartitionSoftwareComponentInstanceRef

Parameter Name	EcucPartitionSoftwareComponentInstanceRef
Parent Container	EcucPartition
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]
Description	References the SW Component instances from the Ecu Extract that shall be executed in this partition.
Multiplicity	0..*
Type	Instance reference to SW-COMPONENT-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE
Pre-Compile only	true

C.1.2 Pdu

[ECUC_EcuC_00001] Definition of EcucParamConfContainerDef Pdu

Container Name	Pdu	
Parent Container	EcucPduCollection	
Upstream mapping defined	–	
Description	One Pdu flowing through the COM-Stack. This Pdu is used by all Com-Stack modules to agree on referencing the same Pdu.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DynamicLength	0..1	[ECUC_EcuC_00078]
J1939Requestable	0..1	[ECUC_EcuC_00072]
KeepLocalPduBuffer	0..1	[ECUC_EcuC_00087]
PduBufferAlignment	0..1	[ECUC_EcuC_00088]
PduLength	1	[ECUC_EcuC_00003]
EcucPduDefaultPartitionRef	0..1	[ECUC_EcuC_00082]
MetaDataTypeRef	0..1	[ECUC_EcuC_00077]
SysTPduToFrameTriggeringRef	0..1	[ECUC_EcuC_00052]
SysTPduToPduTriggeringRef	0..1	[ECUC_EcuC_00054]

Included Containers		
Container Name	Multiplicity	Description
EcucPduDedicatedPartition	0..*	Module specific container for Pdu to partition assignment.

[ECUC_EcuC_00078] Definition of EcucBooleanParamDef DynamicLength

Parameter Name	DynamicLength	
Parent Container	Pdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	This parameter defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_EcuC_00072] Definition of EcucBooleanParamDef J1939Requestable

Parameter Name	J1939Requestable
Parent Container	Pdu
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Pdu can be triggered by the J1939 request message.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–

▽

△

Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_EcuC_00087] Definition of EcucBooleanParamDef KeepLocalPduBuffer

Parameter Name	KeepLocalPduBuffer
Parent Container	Pdu
Upstream mapping defined	–
Description	This parameter defines whether a module that handles the PDU would keep a temporary local buffer • until a confirmation or release-rx-buffer function call arrives (KeepLocalPduBuffer = TRUE) or • if a temporary local buffer is released after transmission or rx indication return (Keep LocalPduBuffer = FALSE). **Tags:** atp.Status=draft
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_EcuC_00088] Definition of EcucIntegerParamDef PduBufferAlignment

Parameter Name	PduBufferAlignment	
Parent Container	Pdu	
Upstream mapping defined	–	
Description	This parameter defines the byte alignment of temporary local buffers that is required by the hardware. Using this parameter, a configuration can ensure that an upper layer module is aware of the alignment requirements of a driver. Unit: Byte **Tags:** atp.Status=draft Dependency: This parameter shall only be configured if KeepLocalPduBuffer is True.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 8	
Default value	2	
Pre-Compile only	true	

[ECUC_EcuC_00003] Definition of EcucIntegerParamDef PduLength

Parameter Name	PduLength
Parent Container	Pdu
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Length of the Pdu in bytes. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits.
Multiplicity	1
Type	EcucIntegerParamDef

▽

△

Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

[ECUC_EcuC_00082] Definition of EcucReferenceDef EcucPduDefaultPartitionRef

Parameter Name	EcucPduDefaultPartitionRef
Parent Container	Pdu
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according Pdu is assigned to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

[ECUC_EcuC_00077] Definition of EcucReferenceDef MetaDataTypeRef

Parameter Name	MetaDataTypeRef
Parent Container	Pdu
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [13, CP_TPS_SystemTemplate]
Description	Reference to meta data that is transported in the Pdu through the AUTOSAR layers.
Multiplicity	0..1
Type	Reference to MetaDataType
Pre-Compile only	true

[ECUC_EcuC_00052] Definition of EcucForeignReferenceDef SysTPduToFrameTriggeringRef

Parameter Name	SysTPduToFrameTriggeringRef	
Parent Container	Pdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Reference to the FrameTriggering from the SystemTemplate which this Pdu belongs to. SysTPduToFrameTriggeringRef shall be used for UserDefinedPdus, NmPdus and NPdus which are not going through the Pdu Router. This reference shall not be used if SysTPduToPduTriggeringRef exists. Dependency: SysTPduToFrameTriggeringRef shall be used for UserDefinedPdus, Nm Pdus and NPdus which are not going through the Pdu Router. This reference shall not be used if SysTPduToPduTriggeringRef exists.	
Multiplicity	0..1	
Type	Foreign reference to FRAME-TRIGGERING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_EcuC_00054] Definition of EcucForeignReferenceDef SysTPduToPduTriggeringRef

Parameter Name	SysTPduToPduTriggeringRef	
Parent Container	Pdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Reference to the PduTriggering from the SystemTemplate which this Pdu represents. SysTPduToPduTriggeringRef shall be used for all Pdus except UserDefinedPdus, Nm Pdus and NPdus which are not going through the Pdu Router. For these Pdus, Sys TPduToFrameTriggeringRef shall be used. Dependency: SysTPduToPduTriggeringRef shall be used for all Pdus except User DefinedPdus, NmPdus and NPdus which are not going through the Pdu Router. This reference shall not be used if SysTPduToFrameTriggeringRef exists.	
Multiplicity	0..1	
Type	Foreign reference to PDU-TRIGGERING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

C.1.3 EcucPduDedicatedPartition

[ECUC_EcuC_00079] Definition of EcucParamConfContainerDef EcucPduDedicatedPartition

Container Name	EcucPduDedicatedPartition	
Parent Container	Pdu	
Upstream mapping defined	–	
Description	Module specific container for Pdu to partition assignment.	
Multiplicity	0..*	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucPduDedicatedPartitionBswModuleRef	1	[ECUC_EcuC_00080]
EcucPduDedicatedPartitionRef	1	[ECUC_EcuC_00081]

No Included Containers

[ECUC_EcuC_00080] Definition of EcucForeignReferenceDef EcucPduDedicatedPartitionBswModuleRef

Parameter Name	EcucPduDedicatedPartitionBswModuleRef
Parent Container	EcucPduDedicatedPartition
Upstream mapping defined	–
Description	Reference to BSW module, for which the according dedicated Pdu assignment is valid.
Multiplicity	1
Type	Foreign reference to ECUC-MODULE-CONFIGURATION-VALUES

▽

△

Pre-Compile only	true

[ECUC_EcuC_00081] Definition of EcucReferenceDef EcucPduDedicatedPartitionRef

Parameter Name	EcucPduDedicatedPartitionRef
Parent Container	EcucPduDedicatedPartition
Upstream mapping defined	–
Description	Module specific reference to EcucPartition, where the according Pdu is assigned to. The dedicated partition reference shall overrule the default partition reference for the respective module.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

C.2 RTE

C.2.1 RteEventToTaskMapping

[ECUC_Rte_09020] Definition of EcucParamConfContainerDef RteEventToTaskMapping

Container Name	RteEventToTaskMapping
Parent Container	RteSwComponentInstance
Upstream mapping defined	–
Description	Maps an instance of a RunnableEntity onto one OsTask based on the activating RTEEvent. In the case of a RunnableEntity executed via a direct or trusted function call this RteEventToTaskMapping is still specified but no RteMappedToTask element is included. The RtePositionInTask parameter is necessary to provide an ordering of events invoked by the same RTE API.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RteActivationOffset	0..1	[ECUC_Rte_09018]
RteEventIsMappedToTask	0..1	[ECUC_Rte_09221]
RteModeSenderQueueLength	0..1	[ECUC_Rte_09228]
RteOsSchedulePoint	0..1	[ECUC_Rte_09022]
RtePositionInTask	0..1	[ECUC_Rte_09023]
RteQueuedReceiverQueueLength	0..1	[ECUC_Rte_09227]
RteServerNumberOfRequestProcessing	0..1	[ECUC_Rte_09214]
RteServerQueueLength	0..1	[ECUC_Rte_09133]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
RteEventPredecessorSyncPointRef	0..1	[ECUC_Rte_09128]
RteEventRef	1..*	[ECUC_Rte_09019]
RteEventSuccessorSyncPointRef	0..1	[ECUC_Rte_09129]
RteMappedToTaskRef	0..1	[ECUC_Rte_09021]
RteRipsFillRoutineRef	0..*	[ECUC_Rte_89005]
RteRipsFlushRoutineRef	0..*	[ECUC_Rte_89006]
RteRipsInvocationHandlerRef	0..1	[ECUC_Rte_89008]
RteUsedOsAlarmRef	0..1	[ECUC_Rte_09024]
RteUsedOsEventRef	0..1	[ECUC_Rte_09025]
RteUsedOsSchTblExpiryPointRef	0..1	[ECUC_Rte_09026]
RteVirtuallyMappedToTaskRef	0..1	[ECUC_Rte_09027]

No Included Containers

[ECUC_Rte_09018] Definition of EcucFloatParamDef RteActivationOffset

Parameter Name	RteActivationOffset	
Parent Container	RteEventToTaskMapping	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]	
Description	Activation offset in seconds.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

[ECUC_Rte_09221] Definition of EcucBooleanParamDef RteEventIsMappedToTask

Parameter Name	RteEventIsMappedToTask
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	Defines whether mapping of the RteEvent to an OsTask within this RteEventToTaskMapping container is intentionally not mapped. false: The RteMappedToTaskRef parameter within RteEventToTaskMapping is not applicable and shall not be provided. true: The RteMappedToTaskRef parameter within RteEventToTaskMapping is applicable and shall be provided.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Rte_09228] Definition of EcucIntegerParamDef RteModeSenderQueueLength

Parameter Name	RteModeSenderQueueLength	
Parent Container	RteEventToTaskMapping	
Upstream mapping defined	–	
Description	Specifies the length of mode queue on the mode manager side. This value overwrites the queueLength specified at the ModeSwitchSenderComSpec.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

For parameter table [ECUC_Rte_09022] RteOsSchedulePoint, see definition below container RteBswEventToTaskMapping.

[ECUC_Rte_09023] Definition of EcucIntegerParamDef RtePositionInTask

Parameter Name	RtePositionInTask	
Parent Container	RteEventToTaskMapping	
Upstream mapping defined	–	
Description	Each RunnableEntity mapped to an OsTask has a specific position within the task execution. For periodic activation this is the order of execution. For event driver activation this is the order of evaluation which actual RunnableEntity has to be executed. In case of direct or trusted function calls this parameter is necessary to provide an ordering of events when several ExecutableEntities are invoked by the same RTE API.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Rte_09227] Definition of EcucIntegerParamDef RteQueuedReceiverQueueLength

Parameter Name	RteQueuedReceiverQueueLength	
Parent Container	RteEventToTaskMapping	
Upstream mapping defined	–	
Description	Specifies the length of the event queue on receiver side. This value overwrites the queueLength specified at the QueuedReceiverComSpec.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Rte_09214] Definition of EcucIntegerParamDef RteServerNumberOfRequestProcessing

Parameter Name	RteServerNumberOfRequestProcessing	
Parent Container	RteEventToTaskMapping	
Upstream mapping defined	–	
Description	Specifies the maximum number of queued requests that can be handled within one Os Task execution.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Rte_09133] Definition of EcucIntegerParamDef RteServerQueueLength

Parameter Name	RteServerQueueLength	
Parent Container	RteEventToTaskMapping	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]	
Description	Specifies the length of the queue for the server call serialization. This value overwrites the queueLength specified at the ServerComSpec.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Rte_09128] Definition of EcucReferenceDef RteEventPredecessorSyncPointRef

Parameter Name	RteEventPredecessorSyncPointRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	The RteEventPredecessorSyncPointRef is necessary to provide a cross core synchronization in case of RteEvents triggered by the same event source but mapped to tasks belonging to different partitions on different cores. The synchronization point must be reached by all referencing RteEvents before the execution in all related tasks is continued. In case of RteEventPredecessorSyncPointRef the RunnableEntity activated by the mapped RteEvent is executed after the synchronization point is passed.
Multiplicity	0..1
Type	Reference to RteSyncPoint
Pre-Compile only	true

[ECUC_Rte_09019] Definition of EcucForeignReferenceDef RteEventRef

Parameter Name	RteEventRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]
Description	Reference to the description of the RTEEvent which is pointing to the RunnableEntity being mapped. This allows a fine grained mapping of RunnableEntites based on the activating RTEEvent.
Multiplicity	1..*
Type	Foreign reference to RTE-EVENT
Pre-Compile only	true

[ECUC_Rte_09129] Definition of EcucReferenceDef RteEventSuccessorSyncPointRef

Parameter Name	RteEventSuccessorSyncPointRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	The RteEventSuccessorSyncPointRef is necessary to provide a cross core synchronization in case of RteEvents triggered by the same event source but mapped to tasks belonging to different partitions on different cores. The synchronization point must be reached by all referencing RteEvents before the execution in all related tasks is continued. In case of RteEventSuccessorSyncPointRef the RunnableEntity activated by the mapped RteEvent is executed before the synchronization point is entered.
Multiplicity	0..1
Type	Reference to RteSyncPoint
Pre-Compile only	true

[ECUC_Rte_09021] Definition of EcucReferenceDef RteMappedToTaskRef

Parameter Name	RteMappedToTaskRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	Reference to the OsTask the RunnableEntity activated by the RteEventRef is mapped to. If no reference to the OsTask is specified the RunnableEntity shall be executed via a direct or trusted function call. The fact that no reference to an OsTask is specified for a RunnableEntity does not necessarily imply that every RTE generator has to support the implementation of this RunnableEntity as a direct or trusted function call. The standard set of use cases for direct or trusted function calls that has to be supported by every RTE generator is explicitly stated as requirements in this document. For further optimization RTE vendors are free to support additional scenarios of direct or trusted function call implementations that are not explicitly required in this document.
Multiplicity	0..1
Type	Reference to OsTask
Pre-Compile only	true

For parameter table [ECUC_Rte_89005] RteRipsFillRoutineRef, see definition below container RteBswEventToTaskMapping.

For parameter table [ECUC_Rte_89006] RteRipsFlushRoutineRef, see definition below container RteBswEventToTaskMapping.

[ECUC_Rte_89008] Definition of EcucUriReferenceDef RteRipsInvocationHandlerRef

Parameter Name	RteRipsInvocationHandlerRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	Reference to a Buffer-Fill Routine implemented by an RTE Implementation Plug-In. This routine gets invoked directly before the ExecutableEntity is started.
Multiplicity	0..1
Type	Reference to destinationUri RteRipsUriDefSet/RteRipsInvocationHandler
Pre-Compile only	true

[ECUC_Rte_09024] Definition of EcucReferenceDef RteUsedOsAlarmRef

Parameter Name	RteUsedOsAlarmRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	If an OsAlarm is used to activate the OsTask this RteEvent is mapped to it shall be referenced here.
Multiplicity	0..1
Type	Reference to OsAlarm
Pre-Compile only	true

[ECUC_Rte_09025] Definition of EcucReferenceDef RteUsedOsEventRef

Parameter Name	RteUsedOsEventRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	If an OsEvent is used to activate the OsTask this RteEvent is mapped to it shall be referenced here.
Multiplicity	0..1
Type	Reference to OsEvent
Pre-Compile only	true

[ECUC_Rte_09026] Definition of EcucReferenceDef RteUsedOsSchTblExpiryPointRef

Parameter Name	RteUsedOsSchTblExpiryPointRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	If an OsScheduleTableExpiryPoint is used to activate the OsTask this RteEvent is mapped to it shall be referenced here.
Multiplicity	0..1
Type	Reference to OsScheduleTableExpiryPoint
Pre-Compile only	true

[ECUC_Rte_09027] Definition of EcucReferenceDef RteVirtuallyMappedToTaskRef

Parameter Name	RteVirtuallyMappedToTaskRef
Parent Container	RteEventToTaskMapping
Upstream mapping defined	–
Description	Optional reference to an OsTask where the activation of this RteEvent shall be evaluated. The actual execution of the Runnable Entity shall happen in the OsTask referenced by RteMappedToTaskRef.
Multiplicity	0..1
Type	Reference to OsTask
Pre-Compile only	true

C.2.2 RteInitializationRunnableBatch

[ECUC_Rte_09115] Definition of EcucParamConfContainerDef RteInitialization RunnableBatch

Container Name	RteInitializationRunnableBatch
Parent Container	Rte
Upstream mapping defined	–
Description	This container corresponds to an Rte_Init_<shortName of this container> function invoking the mapped RunnableEntities.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

No Included Containers

C.3 Os

C.3.1 OsAlarm

[ECUC_Os_00003] Definition of EcucParamConfContainerDef OsAlarm

Container Name	OsAlarm
Parent Container	Os
Upstream mapping defined	–
Description	An OsAlarm may be used to asynchronously inform or activate a specific task. It is possible to start alarms automatically at system start-up depending on the application mode.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsAlarmAccessingApplication	0..*	[ECUC_Os_00004]
OsAlarmCounterRef	1	[ECUC_Os_00005]

Included Containers		
Container Name	**Multiplicity**	**Description**
OsAlarmAction	1	This container defines which type of notification is used when the alarm expires.
OsAlarmAutostart	0..1	If present this container defines if an alarm is started automatically at system start-up depending on the application mode.

[ECUC_Os_00004] Definition of EcucReferenceDef OsAlarmAccessingApplication

Parameter Name	OsAlarmAccessingApplication
Parent Container	OsAlarm
Upstream mapping defined	–
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Pre-Compile only	true

[ECUC_Os_00005] Definition of EcucReferenceDef OsAlarmCounterRef

Parameter Name	OsAlarmCounterRef
Parent Container	OsAlarm
Upstream mapping defined	–
Description	Reference to the assigned counter for that alarm
Multiplicity	1
Type	Reference to OsCounter
Pre-Compile only	true

C.3.2 OsApplication

[ECUC_Os_00114] Definition of EcucParamConfContainerDef OsApplication

Container Name	OsApplication
Parent Container	Os
Upstream mapping defined	–

▽

△

Description	An AUTOSAR OS must be capable of supporting a collection of OS objects (tasks, interrupts, alarms, hooks etc.) that form a cohesive functional unit. This collection of objects is termed an OS-Application. All objects which belong to the same OS-Application have access to each other. Access means to allow to use these objects within API services. Access by other applications can be granted separately.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsTrusted	1	[ECUC_Os_00115]
OsTrustedApplicationDelayTimingViolationCall	1	[ECUC_Os_00395]
OsTrustedApplicationWithProtection	1	[ECUC_Os_00394]
OsAppAlarmRef	0..*	[ECUC_Os_00231]
OsAppCounterRef	0..*	[ECUC_Os_00234]
OsAppEcucPartitionRef	1	[ECUC_Os_00392]
OsAppIsrRef	0..*	[ECUC_Os_00221]
OsAppScheduleTableRef	0..*	[ECUC_Os_00230]
OsAppTaskRef	0..*	[ECUC_Os_00116]
OsMemoryMappingCodeLocationRef	0..1	[ECUC_Os_00402]

Included Containers		
Container Name	**Multiplicity**	**Description**
OsApplicationHooks	1	Container to structure the OS-Application-specific hooks
OsApplicationTrustedFunction	0..*	Container to structure the configuration parameters of trusted functions

[ECUC_Os_00115] Definition of EcucBooleanParamDef OsTrusted

Parameter Name	OsTrusted
Parent Container	OsApplication
Upstream mapping defined	–
Description	Parameter to specify if an OS-Application is trusted or not. true: OS-Application is trusted false: OS-Application is not trusted (default) Dependency: Required for scalability classes 3 and 4.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Os_00395] Definition of EcucBooleanParamDef OsTrustedApplicationDelayTimingViolationCall

Parameter Name	OsTrustedApplicationDelayTimingViolationCall
Parent Container	OsApplication
Upstream mapping defined	–
Description	Parameter to specify if a timing violation which occurs within an trusted OS-Application is raised immediately or if it is delayed until the current task returns to the calling OS-Application (return of CallTrustedFunction) true: violation / call to ProtectionHook() is delayed false: timing violation cause an immediate call to the ProtectionHook().
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

[ECUC_Os_00394] Definition of EcucBooleanParamDef OsTrustedApplicationWithProtection

Parameter Name	OsTrustedApplicationWithProtection
Parent Container	OsApplication
Upstream mapping defined	–
Description	Parameter to specify if a trusted OS-Application is executed with memory protection or not. true: OS-Application runs within a protected environment. This means that write access is limited. false: OS-Application has full write access (default)
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Os_00231] Definition of EcucReferenceDef OsAppAlarmRef

Parameter Name	OsAppAlarmRef
Parent Container	OsApplication
Upstream mapping defined	–
Description	Specifies the OsAlarms that belong to the OsApplication.
Multiplicity	0..*
Type	Reference to OsAlarm
Pre-Compile only	true

[ECUC_Os_00234] Definition of EcucReferenceDef OsAppCounterRef

Parameter Name	OsAppCounterRef
Parent Container	OsApplication
Upstream mapping defined	–
Description	References the OsCounters that belong to the OsApplication.
Multiplicity	0..*

▽

△

Type	Reference to OsCounter
Pre-Compile only	true

[ECUC_Os_00392] Definition of EcucReferenceDef OsAppEcucPartitionRef

Parameter Name	OsAppEcucPartitionRef
Parent Container	OsApplication
Upstream mapping defined	–
Description	Denotes which "EcucPartition" is implemented by this "OSApplication".
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

[ECUC_Os_00221] Definition of EcucReferenceDef OsAppIsrRef

Parameter Name	OsAppIsrRef
Parent Container	OsApplication
Upstream mapping defined	–
Description	references which OsIsrs belong to the OsApplication
Multiplicity	0..*
Type	Reference to OsIsr
Pre-Compile only	true

[ECUC_Os_00230] Definition of EcucReferenceDef OsAppScheduleTableRef

Parameter Name	OsAppScheduleTableRef
Parent Container	OsApplication
Upstream mapping defined	–
Description	References the OsScheduleTables that belong to the OsApplication.
Multiplicity	0..*
Type	Reference to OsScheduleTable
Pre-Compile only	true

[ECUC_Os_00116] Definition of EcucReferenceDef OsAppTaskRef

Parameter Name	OsAppTaskRef
Parent Container	OsApplication
Upstream mapping defined	–
Description	references which OsTasks belong to the OsApplication
Multiplicity	0..*
Type	Reference to OsTask
Pre-Compile only	true

[ECUC_Os_00402] Definition of EcucForeignReferenceDef OsMemoryMapping CodeLocationRef

Parameter Name	OsMemoryMappingCodeLocationRef
Parent Container	OsApplication, OsApplicationHooks, OsHooks, OsIsr, OsTask
Upstream mapping defined	–
Description	Reference to the memory mapping containing details about the section where the code is placed.
Multiplicity	0..1
Type	Foreign reference to SW-ADDR-METHOD
Pre-Compile only	true

C.3.3 OsCounter

[ECUC_Os_00026] Definition of EcucParamConfContainerDef OsCounter

Container Name	OsCounter
Parent Container	Os
Upstream mapping defined	–
Description	Configuration information for the counters that belong to the OsApplication.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsCounterMaxAllowedValue	1	[ECUC_Os_00027]
OsCounterMinCycle	1	[ECUC_Os_00028]
OsCounterTicksPerBase	1	[ECUC_Os_00029]
OsCounterType	1	[ECUC_Os_00255]
OsSecondsPerTick	0..1	[ECUC_Os_00030]
OsCounterAccessingApplication	0..*	[ECUC_Os_00031]

Included Containers		
Container Name	**Multiplicity**	**Description**
OsDriver	0..1	This Container contains the information who will drive the counter. This configuration is only valid if the counter has Os CounterType set to HARDWARE. If the container does not exist (multiplicity=0) the timer is managed by the OS internally (OSINTERNAL). If the container exists the OS can use the GPT interface to manage the timer. The user have to supply the GPT channel. If the counter is driven by some other (external to the OS) source (like a TPU for example) this must be described as a vendor specific extension.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
OsTimeConstant	0..*	Allows the user to define constants which can be e.g. used to compare time values with timer tick values. A time value will be converted to a timer tick value during generation and can later on accessed via the OsConstName. The conversation is done by rounding time values to the nearest fitting tick value.

[ECUC_Os_00027] Definition of EcucIntegerParamDef OsCounterMaxAllowed Value

Parameter Name	OsCounterMaxAllowedValue	
Parent Container	OsCounter	
Upstream mapping defined	–	
Description	Maximum possible allowed value of the system counter in ticks.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 18446744073709551615	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00028] Definition of EcucIntegerParamDef OsCounterMinCycle

Parameter Name	OsCounterMinCycle	
Parent Container	OsCounter	
Upstream mapping defined	–	
Description	The MINCYCLE attribute specifies the minimum allowed number of counter ticks for a cyclic alarm linked to the counter.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 18446744073709551615	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00029] Definition of EcucIntegerParamDef OsCounterTicksPerBase

Parameter Name	OsCounterTicksPerBase	
Parent Container	OsCounter	
Upstream mapping defined	–	
Description	The TICKSPERBASE attribute specifies the number of ticks required to reach a counterspecific unit. The interpretation is implementation-specific.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00255] Definition of EcucEnumerationParamDef OsCounterType

Parameter Name	OsCounterType	
Parent Container	OsCounter	
Upstream mapping defined	–	
Description	This parameter contains the natural type or unit of the counter.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HARDWARE	This counter is driven by some hardware e.g. a hardware timer unit.
	SOFTWARE	The counter is driven by some software which calls the IncrementCounter service.
Pre-Compile only	true	

[ECUC_Os_00030] Definition of EcucFloatParamDef OsSecondsPerTick

Parameter Name	OsSecondsPerTick	
Parent Container	OsCounter	
Upstream mapping defined	–	
Description	Time of one counter tick in seconds.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00031] Definition of EcucReferenceDef OsCounterAccessingApplication

Parameter Name	OsCounterAccessingApplication
Parent Container	OsCounter
Upstream mapping defined	–
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Pre-Compile only	true

C.3.4 OsEvent

[ECUC_Os_00033] Definition of EcucParamConfContainerDef OsEvent

Container Name	OsEvent
Parent Container	Os
Upstream mapping defined	–

▽

△

Description	Representation of OS events in the configuration context. Adopted from the ISO 17356-6 specification.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsEventMask	0..1	[ECUC_Os_00034]

No Included Containers

[ECUC_Os_00034] Definition of EcucIntegerParamDef OsEventMask

Parameter Name	OsEventMask
Parent Container	OsEvent
Upstream mapping defined	–
Description	If event mask would be set to AUTO in OIL, this parameter should be omitted here.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	0 .. 18446744073709551615
Default value	–
Pre-Compile only	true

C.3.5 OsResource

[ECUC_Os_00252] Definition of EcucParamConfContainerDef OsResource

Container Name	OsResource
Parent Container	Os
Upstream mapping defined	–
Description	An OsResource object is used to co-ordinate the concurrent access by tasks and ISRs to a shared resource, e.g. the scheduler, any program sequence, memory or any hardware area.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsResourceProperty	1	[ECUC_Os_00050]
OsResourceAccessingApplication	0..*	[ECUC_Os_00051]
OsResourceLinkedResourceRef	0..1	[ECUC_Os_00052]

No Included Containers

[ECUC_Os_00050] Definition of EcucEnumerationParamDef OsResourceProperty

Parameter Name	OsResourceProperty	
Parent Container	OsResource	
Upstream mapping defined	–	
Description	This specifies the type of the resource.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	INTERNAL	The resource is an internal resource.
	LINKED	The resource is a linked resource (a second name for a existing resource).
	STANDARD	The resource is a standard resource.
Pre-Compile only	true	

[ECUC_Os_00051] Definition of EcucReferenceDef OsResourceAccessingApplication

Parameter Name	OsResourceAccessingApplication
Parent Container	OsResource
Upstream mapping defined	–
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Pre-Compile only	true

[ECUC_Os_00052] Definition of EcucReferenceDef OsResourceLinkedResourceRef

Parameter Name	OsResourceLinkedResourceRef
Parent Container	OsResource
Upstream mapping defined	–
Description	The link to the resource. Must be valid if OsResourceProperty is LINKED. If Os ResourceProperty is not LINKED the value is ignored.
Multiplicity	0..1
Type	Reference to OsResource
Pre-Compile only	true

C.3.6 OsScheduleTable

[ECUC_Os_00141] Definition of EcucParamConfContainerDef OsScheduleTable

Container Name	OsScheduleTable
Parent Container	Os
Upstream mapping defined	–

▽

△

Description	An OsScheduleTable addresses the synchronization issue by providing an encapsulation of a statically defined set of alarms that cannot be modified at runtime.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsScheduleTableDuration	1	[ECUC_Os_00053]
OsScheduleTableRepeating	1	[ECUC_Os_00144]
OsScheduleTableCounterRef	1	[ECUC_Os_00145]
OsSchTblAccessingApplication	0..*	[ECUC_Os_00054]

Included Containers		
Container Name	**Multiplicity**	**Description**
OsScheduleTableAutostart	0..1	This container specifies if and how the schedule table is started on startup of the Operating System. The options to start a schedule table correspond to the API calls to start schedule tables during runtime.
OsScheduleTableExpiryPoint	1..*	The point on a Schedule Table at which the OS activates tasks and/or sets events
OsScheduleTableSync	0..1	This container specifies the synchronization parameters of the schedule table.

[ECUC_Os_00053] Definition of EcucIntegerParamDef OsScheduleTableDuration

Parameter Name	OsScheduleTableDuration
Parent Container	OsScheduleTable
Upstream mapping defined	–
Description	This parameter defines the modulus of the schedule table (in ticks).
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 18446744073709551615
Default value	–
Pre-Compile only	true

[ECUC_Os_00144] Definition of EcucBooleanParamDef OsScheduleTableRepeating

Parameter Name	OsScheduleTableRepeating
Parent Container	OsScheduleTable
Upstream mapping defined	–
Description	true: first expiry point on the schedule table shall be processed at final expiry point delay ticks after the final expiry point is processed. false: the schedule table processing stops when the final expiry point is processed.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	–
Pre-Compile only	true

[ECUC_Os_00145] Definition of EcucReferenceDef OsScheduleTableCounterRef

Parameter Name	OsScheduleTableCounterRef
Parent Container	OsScheduleTable
Upstream mapping defined	–
Description	This parameter contains a reference to the counter which drives the schedule table.
Multiplicity	1
Type	Reference to OsCounter
Pre-Compile only	true

[ECUC_Os_00054] Definition of EcucReferenceDef OsSchTblAccessingApplication

Parameter Name	OsSchTblAccessingApplication
Parent Container	OsScheduleTable
Upstream mapping defined	–
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Pre-Compile only	true

C.3.7 OsScheduleTableExpiryPoint

[ECUC_Os_00143] Definition of EcucParamConfContainerDef OsScheduleTableExpiryPoint

Container Name	OsScheduleTableExpiryPoint
Parent Container	OsScheduleTable
Upstream mapping defined	–
Description	The point on a Schedule Table at which the OS activates tasks and/or sets events
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsScheduleTblExpPointOffset	1	[ECUC_Os_00062]

Included Containers		
Container Name	**Multiplicity**	**Description**
OsScheduleTableEventSetting	0..*	Event that is triggered by that schedule table.
OsScheduleTableTaskActivation	0..*	Task that is triggered by that schedule table.
OsScheduleTableTaskReplenish	0..*	Task that is replenished by that expiry point
OsScheduleTblAdjustableExpPoint	0..1	Adjustable expiry point

[ECUC_Os_00062] Definition of EcucIntegerParamDef OsScheduleTblExpPoint Offset

Parameter Name	OsScheduleTblExpPointOffset	
Parent Container	OsScheduleTableExpiryPoint	
Upstream mapping defined	–	
Description	The offset from zero (in ticks) at which the expiry point is to be processed.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Pre-Compile only	true	

C.3.8 OsSpinlock

[ECUC_Os_00258] Definition of EcucParamConfContainerDef OsSpinlock

Container Name	OsSpinlock
Parent Container	Os
Upstream mapping defined	–
Description	An OsSpinlock object is used to co-ordinate concurrent access by TASKs/ISR2s on different cores to a shared resource.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsSpinlockLockMethod	1	[ECUC_Os_01038]
OsSpinlockAccessingApplication	1..*	[ECUC_Os_01021]
OsSpinlockSuccessor	0..1	[ECUC_Os_01022]

No Included Containers

[ECUC_Os_01038] Definition of EcucEnumerationParamDef OsSpinlockLockMethod

Parameter Name	OsSpinlockLockMethod	
Parent Container	OsSpinlock	
Upstream mapping defined	–	
Description	Lock method which is used when a spinlock is taken. Note that it is possible that a user (e.g. a Task) might hold more than one spinlock. In this case the last lock taken is forced to use at least a lock methode which locks as strong as the current one.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	LOCK_ALL_INTERRUPTS	Also disable all interrupts.
	LOCK_CAT2_INTERRUPTS	Also disable category 2 interrupts (under Os control).
	LOCK_NOTHING	No additional action.
	LOCK_WITH_RES_SCHEDULER	Also lock the scheduler OsResource
Default value	LOCK_NOTHING	
Pre-Compile only	true	

[ECUC_Os_01021] Definition of EcucReferenceDef OsSpinlockAccessingApplication

Parameter Name	OsSpinlockAccessingApplication
Parent Container	OsSpinlock
Upstream mapping defined	–
Description	Reference to OsApplications that have an access to this object.
Multiplicity	1..*
Type	Reference to OsApplication
Pre-Compile only	true

[ECUC_Os_01022] Definition of EcucReferenceDef OsSpinlockSuccessor

Parameter Name	OsSpinlockSuccessor
Parent Container	OsSpinlock
Upstream mapping defined	–
Description	To check whether a spinlock can be occupied (in a nested way) without any danger of deadlock, a linked list of spinlocks can be defined. A spinlock can only be occupied in the order of the linked list. It is allowed to skip a spinlock. If no linked list is specified, spinlocks cannot be nested.
Multiplicity	0..1
Type	Reference to OsSpinlock
Pre-Compile only	true

C.3.9 OsTask

[ECUC_Os_00073] Definition of EcucParamConfContainerDef OsTask

Container Name	OsTask
Parent Container	Os
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This container represents an ISO 17356 task.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OsTaskActivation	1	[ECUC_Os_00074]
OsTaskPeriod	0..1	[ECUC_Os_00404]
OsTaskPriority	1	[ECUC_Os_00075]
OsTaskSchedule	1	[ECUC_Os_00076]
OsMemoryMappingCodeLocationRef	0..1	[ECUC_Os_00402]
OsTaskAccessingApplication	0..*	[ECUC_Os_00077]
OsTaskEventRef	0..*	[ECUC_Os_00078]
OsTaskResourceRef	0..*	[ECUC_Os_00079]

Included Containers		
Container Name	**Multiplicity**	**Description**
OsTaskAutostart	0..1	This container determines whether the task is activated during the system start-up procedure or not for some specific application modes. If the task shall be activated during the system start-up, this container is present and holds the references to the application modes in which the task is auto-started.
OsTaskTimingProtection	0..1	This container contains all parameters regarding timing protection of the task.

[ECUC_Os_00074] Definition of EcucIntegerParamDef OsTaskActivation

Parameter Name	OsTaskActivation	
Parent Container	OsTask	
Upstream mapping defined	–	
Description	This attribute defines the maximum number of queued activation requests for the task. A value equal to "1" means that at any time only a single activation is permitted for this task. Note that the value must be a natural number starting at 1.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00404] Definition of EcucFloatParamDef OsTaskPeriod

Parameter Name	OsTaskPeriod	
Parent Container	OsTask	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	This parameter specifies the period in seconds of this task in case of a cyclically activated task. If this parameter is not given the task can be activated sporadicly or cyclically with a unknown period value. This value is information, e.g. for time base calculations in the RTE in case Timing Events are mapped onto this OsTask.Be aware, that this parameter is not supposed to be relevant for the OS! This information is given as part of the OS configuration to support configuration work flows using a fixed set of OsTasks.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[-INF .. INF]	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00075] Definition of EcucIntegerParamDef OsTaskPriority

Parameter Name	OsTaskPriority	
Parent Container	OsTask	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The priority of a task is defined by the value of this attribute. This value has to be understood as a relative value, i.e. the values show only the relative ordering of the tasks. ISO 17356-3 defines the lowest priority as zero (0); larger values correspond to higher priorities.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

[ECUC_Os_00076] Definition of EcucEnumerationParamDef OsTaskSchedule

Parameter Name	OsTaskSchedule	
Parent Container	OsTask	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The OsTaskSchedule attribute defines the preemptability of the task. If this attribute is set to NON, no internal resources may be assigned to this task.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FULL	Task is preemptable.
	NON	Task is not preemptable.
Pre-Compile only	true	

For parameter table [ECUC_Os_00402] OsMemoryMappingCodeLocationRef, see definition below container OsApplication.

[ECUC_Os_00077] Definition of EcucReferenceDef OsTaskAccessingApplication

Parameter Name	OsTaskAccessingApplication
Parent Container	OsTask
Upstream mapping defined	–
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Pre-Compile only	true

[ECUC_Os_00078] Definition of EcucReferenceDef OsTaskEventRef

Parameter Name	OsTaskEventRef
Parent Container	OsTask
Upstream mapping defined	–
Description	This reference defines the list of events the extended task may react on.
Multiplicity	0..*
Type	Reference to OsEvent
Pre-Compile only	true

[ECUC_Os_00079] Definition of EcucReferenceDef OsTaskResourceRef

Parameter Name	OsTaskResourceRef
Parent Container	OsTask
Upstream mapping defined	–
Description	This reference defines a list of resources accessed by this task.
Multiplicity	0..*
Type	Reference to OsResource
Pre-Compile only	true

C.4 NvM

C.4.1 NvMBlockDescriptor

[ECUC_NvM_00061] Definition of EcucParamConfContainerDef NvMBlockDescriptor

Container Name	NvMBlockDescriptor
Parent Container	NvM
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate]

▽

△

Description	Container for a management structure to configure the composition of a given NVRAM Block Management Type. Its multiplicity describes the number of configured NVRAM blocks, one block is required to be configured. The NVRAM block descriptors are condensed in the NVRAM block descriptor table.
Multiplicity	1..65536
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
NvMBlockCrcType	0..1	[ECUC_NvM_00476]
NvMBlockHeaderInclude	0..1	[ECUC_NvM_00554]
NvMBlockJobPriority	1	[ECUC_NvM_00477]
NvMBlockManagementType	1	[ECUC_NvM_00062]
NvMBlockUseAutoValidation	1	[ECUC_NvM_00557]
NvMBlockUseCompression	1	[ECUC_NvM_00563]
NvMBlockUseCrc	1	[ECUC_NvM_00036]
NvMBlockUseCRCCompMechanism	1	[ECUC_NvM_00556]
NvMBlockUsePort	1	[ECUC_NvM_00559]
NvMBlockUseSetRamBlockStatus	1	[ECUC_NvM_00552]
NvMBlockUseSyncMechanism	1	[ECUC_NvM_00519]
NvMBlockWriteProt	1	[ECUC_NvM_00033]
NvMBswMBlockStatusInformation	1	[ECUC_NvM_00551]
NvMCalcRamBlockCrc	0..1	[ECUC_NvM_00119]
NvMMaxNumOfReadRetries	1	[ECUC_NvM_00533]
NvMMaxNumOfWriteRetries	1	[ECUC_NvM_00499]
NvMNvBlockBaseNumber	1	[ECUC_NvM_00478]
NvMNvBlockLength	1	[ECUC_NvM_00479]
NvMNvBlockNum	1	[ECUC_NvM_00480]
NvMNvramBlockIdentifier	1	[ECUC_NvM_00481]
NvMNvramDeviceId	1	[ECUC_NvM_00035]
NvMRamBlockDataAddress	0..1	[ECUC_NvM_00482]
NvMReadRamBlockFromNvCallback	0..1	[ECUC_NvM_00521]
NvMResistantToChangedSw	1	[ECUC_NvM_00483]
NvMRomBlockDataAddress	0..1	[ECUC_NvM_00484]
NvMRomBlockNum	1	[ECUC_NvM_00485]
NvMSelectBlockForFirstInitAll	0..1	[ECUC_NvM_00558]
NvMSelectBlockForReadAll	0..1	[ECUC_NvM_00117]
NvMSelectBlockForWriteAll	0..1	[ECUC_NvM_00549]
NvMStaticBlockIDCheck	1	[ECUC_NvM_00532]
NvMWriteBlockOnce	1	[ECUC_NvM_00072]
NvMWriteRamBlockToNvCallback	0..1	[ECUC_NvM_00520]
NvMWriteVerification	1	[ECUC_NvM_00534]
NvMWriteVerificationDataSize	1	[ECUC_NvM_00538]
NvMBlockCipheringRef	0..1	[ECUC_NvM_00567]
NvMBlockEcucPartitionRef	1	[ECUC_NvM_00564]

Included Containers		
Container Name	**Multiplicity**	**Description**
NvMInitBlockCallback	0..1	The presence of this container indicates, that a block specific callback routine is called if no ROM data is available for initialization of the NVRAM block. If the container is not present, no callback routine is called for initialization of the NVRAM block with default data. In case the container has a NvMInitBlockCallbackFnc, the NvM will call this function. In case there is no NvMInitBlockCallbackFnc, the NvM will have an port PNIB_{Block}.
NvMSingleBlockCallback	0..1	The presence of this container indicates, that the block specific callback routine which shall be invoked on termination of each asynchronous single block request [SWS_NvM_00113] If the container is not present, no callback routine is called.. In case the container has a NvMSingleBlockCallbackFnc, the Nv M will call this function. In case there is no NvMSingleBlockCallbackFnc, the NvM will have an port PNJF_{Block}.
NvMTargetBlockReference	1	This parameter is just a container for the parameters for EA and FEE

[ECUC_NvM_00476] Definition of EcucEnumerationParamDef NvMBlockCrcType

Parameter Name	NvMBlockCrcType	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	–	
Description	Defines CRC data width for the NVRAM block. Default: NVM_CRC16, i.e. CRC16 will be used if NVM_BLOCK_USE_CRC==true Dependency: NVM_BLOCK_USE_CRC, NVM_CALC_RAM_BLOCK_CRC	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	NVM_CRC16	(Default) CRC16 will be used if NVM_BLOCK_USE_CRC==true.
	NVM_CRC32	CRC32 is selected for this NVRAM block if NVM_BLOCK_USE_CRC==true.
	NVM_CRC8	CRC8 is selected for this NVRAM block if NVM_BLOCK_USE_CRC==true.
Pre-Compile only	true	

[ECUC_NvM_00554] Definition of EcucStringParamDef NvMBlockHeaderInclude

Parameter Name	NvMBlockHeaderInclude
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines the header file where the owner of the NVRAM block has the declarations of the permanent RAM data block, ROM data block (if configured) and the callback function prototype for each configured callback. If no permanent RAM block, ROM block or callback functions are configured then this configuration parameter shall be ignored.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–

▽

△

Regular Expression	–
Pre-Compile only	true

[ECUC_NvM_00477] Definition of EcucIntegerParamDef NvMBlockJobPriority

Parameter Name	NvMBlockJobPriority	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]	
Description	Defines the job priority for a NVRAM block (0 = Immediate priority).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00062] Definition of EcucEnumerationParamDef NvMBlockManagementType

Parameter Name	NvMBlockManagementType	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]	
Description	Defines the block management type for the NVRAM block.[SWS_NvM_00137]	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	NVM_BLOCK_DATASET	NVRAM block is configured to be of dataset type.
	NVM_BLOCK_NATIVE	NVRAM block is configured to be of native type.
	NVM_BLOCK_REDUNDANT	NVRAM block is configured to be of redundant type.
Pre-Compile only	true	

[ECUC_NvM_00557] Definition of EcucBooleanParamDef NvMBlockUseAutoValidation

Parameter Name	NvMBlockUseAutoValidation
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines whether the RAM Block shall be auto validated during shutdown phase. true: if auto validation mechanism is used, false: otherwise
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00563] Definition of EcucBooleanParamDef NvMBlockUseCompression

Parameter Name	NvMBlockUseCompression
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines whether the data is compressed before written. true: data compression activated (takes more time to read and write) false: no compression **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00036] Definition of EcucBooleanParamDef NvMBlockUseCrc

Parameter Name	NvMBlockUseCrc
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines CRC usage for the NVRAM block, i.e. memory space for CRC is reserved in RAM and NV memory. true: CRC will be used for this NVRAM block. false: CRC will not be used for this NVRAM block. Note: Configuring CRC for a block with immediate priority is not recommended, since the CRC calculation may extend over more than one NvM main function and this could increase the time of writing the immediate data significantly, thus defeating the purpose of immediate priority.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00556] Definition of EcucBooleanParamDef NvMBlockUseCRCCompMechanism

Parameter Name	NvMBlockUseCRCCompMechanism
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines whether the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job. true: if compare mechanism is used, false: otherwise Dependency: False if NvMBlockUseCrc = False
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00559] Definition of EcucBooleanParamDef NvMBlockUsePort

Parameter Name	NvMBlockUsePort
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	If this parameter is true it defines whether: • the port with interface 'NvMMirror' for synchronization mechanism callbacks are generated if the parameter NvMBlockUseSyncMechanism is configured TRUE; • the port with interface 'NvMNotifyInitBlock' for initialization block callback is generated if NvMInitBlockCallback parameter is configured (independent of the content); • the port with interface 'NvMNotifyJobFinished' for single block callback is generated if NvMSingleBlockCallback parameter is configured (independent of the content); • the port with interface 'NvMAdmin' for SetBlockProtection operation is generated.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00552] Definition of EcucBooleanParamDef NvMBlockUseSetRamBlockStatus

Parameter Name	NvMBlockUseSetRamBlockStatus
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines if NvMSetRamBlockStatusApi shall be used for this block or not. Note: If NvMSetRamBlockStatusApi is disabled this configuration parameter shall be ignored. true: calling of NvMSetRamBlockStatus for this RAM block shall set the status of the RAM block. false: calling of NvMSetRamBlockStatus for this RAM block shall be ignored.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00519] Definition of EcucBooleanParamDef NvMBlockUseSyncMechanism

Parameter Name	NvMBlockUseSyncMechanism
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines whether an explicit synchronization mechanism with a RAM mirror and callback routines for transferring data to and from NvM module's RAM mirror is used for NV block. true if synchronization mechanism is used, false otherwise.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00033] Definition of EcucBooleanParamDef NvMBlockWriteProt

Parameter Name	NvMBlockWriteProt
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines an initial write protection of the NV block true: Initial block write protection is enabled. false: Initial block write protection is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00551] Definition of EcucBooleanParamDef NvMBswMBlockStatus Information

Parameter Name	NvMBswMBlockStatusInformation
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	This parameter specifies whether BswM is informed about the current status of the specified block. True: Call BswM_NvM_CurrentBlockMode on changes False: Don't inform BswM at all
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00119] Definition of EcucBooleanParamDef NvMCalcRamBlockCrc

Parameter Name	NvMCalcRamBlockCrc
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines CRC (re)calculation for the permanent RAM block or NVRAM blocks which are configured to use explicit synchronization mechanism. true: CRC will be (re)calculated for this permanent RAM block. false: CRC will not be (re)calculated for this permanent RAM block. Dependency: NVM_BLOCK_USE_CRC
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00533] Definition of EcucIntegerParamDef NvMMaxNumOfReadRetries

Parameter Name	NvMMaxNumOfReadRetries	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	–	
Description	Defines the maximum number of read retries.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	0	
Pre-Compile only	true	

[ECUC_NvM_00499] Definition of EcucIntegerParamDef NvMMaxNumOfWriteRetries

Parameter Name	NvMMaxNumOfWriteRetries	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	–	
Description	Defines the maximum number of write retries for a NVRAM block with [ECUC_NvM_00061]. Regardless of configuration a consistency check (and maybe write retries) are always forced for each block which is processed by the request NvM_WriteAll and NvM_WriteBlock.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00478] Definition of EcucIntegerParamDef NvMNvBlockBaseNumber

Parameter Name	NvMNvBlockBaseNumber	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	–	
Description	Configuration parameter to perform the link between the NVM_NVRAM_BLOCK_IDENTIFIER used by the SW-Cs and the FEE_BLOCK_NUMBER expected by the memory abstraction modules. The parameter value equals the FEE_BLOCK_NUMBER or EA_BLOCK_NUMBER shifted to the right by NvMDatasetSelectionBits bits. (ref. to chapter 7.1.2.1). Calculation Formula: value = TargetBlockReference.[Ea/Fee]BlockConfiguration.[Ea/Fee]BlockNumber >> NvMDatasetSelectionBits Dependency: FEE_BLOCK_NUMBER, EA_BLOCK_NUMBER	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65534	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00479] Definition of EcucIntegerParamDef NvMNvBlockLength

Parameter Name	NvMNvBlockLength	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	–	
Description	Defines the NV block data length in bytes. Note: The implementer can add the attribute 'withAuto' to the parameter definition which indicates that the length can be calculated by the generator automatically (e.g. by using a parser that searches and analyzes the data structure corresponding to the block). When 'withAuto' is set to 'true' for this parameter definition the 'isAutoValue' can be set to 'true'. If 'isAutoValue' is set to 'true' the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00480] Definition of EcucIntegerParamDef NvMNvBlockNum

Parameter Name	NvMNvBlockNum	
Parent Container	NvMBlockDescriptor	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]	
Description	Defines the number of multiple NV blocks in a contiguous area according to the given block management type. 1-255 For NVRAM blocks to be configured of block management type NVM_BLOCK_DATASET. The actual range is limited according to SWS_NvM_00444. 1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE 2 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT Dependency: NVM_BLOCK_MANAGEMENT_TYPE	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00481] Definition of EcucIntegerParamDef NvMNvramBlockIdentifier

Parameter Name	NvMNvramBlockIdentifier
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–

▽

△

Description	Identification of a NVRAM block via a unique block identifier. Implementation Type: NvM_BlockIdType. min = 2 max = 2^(16- NVM_DATASET_SELECTION_BITS)-1 Reserved NVRAM block IDs: 0 -> to derive multi block request results via NvM_Get ErrorStatus 1 -> redundant NVRAM block which holds the configuration ID (generation tool should check that this block is correctly configured from type,CRC and size point of view) Dependency: NVM_DATASET_SELECTION_BITS
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	2 .. 65535
Default value	–
Pre-Compile only	true

[ECUC_NvM_00035] Definition of EcucIntegerParamDef NvMNvramDeviceId

Parameter Name	NvMNvramDeviceId
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines the NVRAM device ID where the NVRAM block is located. Calculation Formula: value = TargetBlockReference.[Ea/Fee]DeviceIndex Dependency: EA_DEVICE_INDEX, FEE_DEVICE_INDEX
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 1
Default value	–
Pre-Compile only	true

[ECUC_NvM_00482] Definition of EcucStringParamDef NvMRamBlockDataAddress

Parameter Name	NvMRamBlockDataAddress
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines the start address of the RAM block data. If this is not configured, no permanent RAM data block is available for the selected block management type.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_NvM_00521] Definition of EcucFunctionNameDef NvMReadRamBlockFromNvCallback

Parameter Name	NvMReadRamBlockFromNvCallback
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Entry address of a block specific callback routine which shall be called in order to let the application copy data from the NvM module's mirror to RAM block. Implementation type: Std_ReturnType E_OK: copy was successful E_NOT_OK: copy was not successful, callback routine to be called again
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_NvM_00483] Definition of EcucBooleanParamDef NvMResistantToChangedSw

Parameter Name	NvMResistantToChangedSw
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines whether a NVRAM block shall be treated resistant to configuration changes or not. If there is no default data available at configuration time then the application shall be responsible for providing the default initialization data. In this case the application has to use NvM_GetErrorStatus()to be able to distinguish between first initialization and corrupted data. true: NVRAM block is resistant to changed software. false: NVRAM block is not resistant to changed software.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00484] Definition of EcucStringParamDef NvMRomBlockDataAddress

Parameter Name	NvMRomBlockDataAddress
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines the start address of the ROM block data. If not configured, no ROM block is available for the selected block management type.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_NvM_00485] Definition of EcucIntegerParamDef NvMRomBlockNum

Parameter Name	NvMRomBlockNum
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines the number of multiple ROM blocks in a contiguous area according to the given block management type. 0-254 For NVRAM blocks to be configured of block management type NVM_BLOCK_DATASET. The actual range is limited according to SWS_NvM_00444. 0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE 0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT Dependency: NVM_BLOCK_MANAGEMENT_TYPE, NVM_NV_BLOCK_NUM
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 254
Default value	–
Pre-Compile only	true

[ECUC_NvM_00558] Definition of EcucBooleanParamDef NvMSelectBlockForFirstInitAll

Parameter Name	NvMSelectBlockForFirstInitAll
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]
Description	Defines whether a block will be processed or not by NvM_FirstInitAll. A block can be configured to be processed even if it doesn't have permanent RAM and/or explicit synchronization. TRUE: block will be processed by NvM_FirstInitAll FALSE: block will not be processed by NvM_FirstInitAll
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00117] Definition of EcucBooleanParamDef NvMSelectBlockForReadAll

Parameter Name	NvMSelectBlockForReadAll
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]

▽

△

Description	Defines whether a NVRAM block shall be processed during NvM_ReadAll or not. This configuration parameter has only influence on those NVRAM blocks which are configured to have a permanent RAM block or which are configured to use explicit synchronization mechanism. true: NVRAM block shall be processed by NvM_ReadAll false: NVRAM block shall not be processed by NvM_ReadAll Dependency: NVM_RAM_BLOCK_DATA_ADDRESS
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00549] Definition of EcucBooleanParamDef NvMSelectBlockForWriteAll

Parameter Name	NvMSelectBlockForWriteAll
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines whether a NVRAM block shall be processed during NvM_WriteAll or not. This configuration parameter has only influence on those NVRAM blocks which are configured to have a permanent RAM block or which are configured to use explicit synchronization mechanism. true: NVRAM block shall be processed by NvM_WriteAll false: NVRAM block shall not be processed by NvM_WriteAll Dependency: NVM_RAM_BLOCK_DATA_ADDRESS
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00532] Definition of EcucBooleanParamDef NvMStaticBlockIDCheck

Parameter Name	NvMStaticBlockIDCheck
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines if the Static Block ID check is enabled. false: Static Block ID check is disabled. true: Static Block ID check is enabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00072] Definition of EcucBooleanParamDef NvMWriteBlockOnce

Parameter Name	NvMWriteBlockOnce
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. true: Defines write protection after first write is enabled. false: Defines write protection after first write is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00520] Definition of EcucFunctionNameDef NvMWriteRamBlockToNvCallback

Parameter Name	NvMWriteRamBlockToNvCallback
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Entry address of a block specific callback routine which shall be called in order to let the application copy data from RAM block to NvM module's mirror. Implementation type: Std_ReturnType E_OK: copy was successful E_NOT_OK: copy was not successful, callback routine to be called again
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_NvM_00534] Definition of EcucBooleanParamDef NvMWriteVerification

Parameter Name	NvMWriteVerification
Parent Container	NvMBlockDescriptor
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [22, CP_TPS_BSWModuleDescription Template]
Description	Defines if Write Verification is enabled. false: Write verification is disabled. true: Write Verification is enabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_NvM_00538] Definition of EcucIntegerParamDef NvMWriteVerificationDataSize

Parameter Name	NvMWriteVerificationDataSize
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Defines the number of bytes to compare in each step when comparing the content of a RAM Block and a block read back.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 65535
Default value	–
Pre-Compile only	true

[ECUC_NvM_00567] Definition of EcucReferenceDef NvMBlockCipheringRef

Parameter Name	NvMBlockCipheringRef
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Reference to ciphering container. If configured, NvM encrypt the data before storage and decrypt the data after restoring. If empty, the NvM stores and restore the original user data. **Tags:** atp.Status=draft Dependency: Key will be located in RAM if this configuration item is not present.
Multiplicity	0..1
Type	Reference to NvMBlockCiphering
Pre-Compile only	true

[ECUC_NvM_00564] Definition of EcucReferenceDef NvMBlockEcucPartitionRef

Parameter Name	NvMBlockEcucPartitionRef
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	Maps the NV block to zero or one ECUC partition to limit the access to this NV block. The ECUC partition referenced is within the subset of the ECUC partitions where the NvM is mapped to.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

C.4.2 NvMInitBlockCallback

[ECUC_NvM_00561] Definition of EcucParamConfContainerDef NvMInitBlockCallback

Container Name	NvMInitBlockCallback
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–
Description	The presence of this container indicates, that a block specific callback routine is called if no ROM data is available for initialization of the NVRAM block. If the container is not present, no callback routine is called for initialization of the NVRAM block with default data. In case the container has a NvMInitBlockCallbackFnc, the NvM will call this function. In case there is no NvMInitBlockCallbackFnc, the NvM will have an port PNIB_{Block}.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
NvMInitBlockCallbackFnc	0..1	[ECUC_NvM_00116]

No Included Containers

[ECUC_NvM_00116] Definition of EcucFunctionNameDef NvMInitBlockCallbackFnc

Parameter Name	NvMInitBlockCallbackFnc
Parent Container	NvMInitBlockCallback
Upstream mapping defined	–
Description	Entry address of a block specific callback routine which shall be called if no ROM data is available for initialization of the NVRAM block. If not configured, no specific callback routine shall be called for initialization of the NVRAM block with default data.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

C.4.3 NvMSingleBlockCallback

[ECUC_NvM_00562] Definition of EcucParamConfContainerDef NvMSingleBlockCallback

Container Name	NvMSingleBlockCallback
Parent Container	NvMBlockDescriptor
Upstream mapping defined	–

▽

△

Description	The presence of this container indicates, that the block specific callback routine which shall be invoked on termination of each asynchronous single block request [SWS_NvM_00113] If the container is not present, no callback routine is called.. In case the container has a NvMSingleBlockCallbackFnc, the NvM will call this function. In case there is no NvMSingleBlockCallbackFnc, the NvM will have an port PNJF_{Block}.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
NvMSingleBlockCallbackFnc	0..1	[ECUC_NvM_00506]

No Included Containers

[ECUC_NvM_00506] Definition of EcucFunctionNameDef NvMSingleBlockCallbackFnc

Parameter Name	NvMSingleBlockCallbackFnc
Parent Container	NvMSingleBlockCallback
Upstream mapping defined	–
Description	Entry address of the block specific callback routine which shall be invoked on termination of each asynchronous single block request [SWS_NvM_00113].
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

C.4.4 NvMCommon

[ECUC_NvM_00028] Definition of EcucParamConfContainerDef NvMCommon

Container Name	NvMCommon
Parent Container	NvM
Upstream mapping defined	–
Description	Container for common configuration options.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
NvMApiConfigClass	1	[ECUC_NvM_00491]
NvMBswMMultiBlockJobStatusInformation	1	[ECUC_NvM_00550]
NvMCompiledConfigId	1	[ECUC_NvM_00492]
NvMCrcNumOfBytes	1	[ECUC_NvM_00493]
NvMCsmRetryCounter	0..1	[ECUC_NvM_00572]
NvMDatasetSelectionBits	1	[ECUC_NvM_00494]
NvMDevErrorDetect	1	[ECUC_NvM_00495]
NvMDynamicConfiguration	1	[ECUC_NvM_00497]
NvMJobPrioritization	1	[ECUC_NvM_00498]
NvMMainFunctionPeriod	1	[ECUC_NvM_00555]
NvMMultiBlockCallback	0..1	[ECUC_NvM_00500]
NvMPollingMode	1	[ECUC_NvM_00501]
NvMRepeatMirrorOperations	1	[ECUC_NvM_00518]
NvMSetRamBlockStatusApi	1	[ECUC_NvM_00502]
NvMSizeImmediateJobQueue	0..1	[ECUC_NvM_00503]
NvMSizeStandardJobQueue	1	[ECUC_NvM_00504]
NvMBufferAlignmentValue	1	[ECUC_NvM_00573]
NvMEcucPartitionRef	1..*	[ECUC_NvM_00565]
NvMMasterEcucPartitionRef	0..1	[ECUC_NvM_00566]

No Included Containers

[ECUC_NvM_00491] Definition of EcucEnumerationParamDef NvMApiConfigClass

Parameter Name	NvMApiConfigClass	
Parent Container	NvMCommon	
Upstream mapping defined	–	
Description	Preprocessor switch to enable some API calls which are related to NVM API configuration classes.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	NVM_API_CONFIG_CLASS_1	All API calls belonging to configuration class 1 are available.
	NVM_API_CONFIG_CLASS_2	All API calls belonging to configuration class 2 are available.
	NVM_API_CONFIG_CLASS_3	All API calls belonging to configuration class 3 are available.
Pre-Compile only	true	

[ECUC_NvM_00550] Definition of EcucBooleanParamDef NvMBswMMultiBlockJobStatusInformation

Parameter Name	NvMBswMMultiBlockJobStatusInformation
Parent Container	NvMCommon
Upstream mapping defined	–

▽

△

Description	This parameter specifies whether BswM is informed about the current status of the multiblock job. True: call BswM_NvM_CurrentJobMode if ReadAll and WriteAll are started, finished, canceled False: do not inform BswM at all
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

[ECUC_NvM_00492] Definition of EcucIntegerParamDef NvMCompiledConfigId

Parameter Name	NvMCompiledConfigId	
Parent Container	NvMCommon	
Upstream mapping defined	–	
Description	Configuration ID regarding the NV memory layout. This configuration ID shall be published as e.g. a SW-C shall have the possibility to write it to NV memory.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00493] Definition of EcucIntegerParamDef NvMCrcNumOfBytes

Parameter Name	NvMCrcNumOfBytes	
Parent Container	NvMCommon	
Upstream mapping defined	–	
Description	If CRC is configured for at least one NVRAM block, this parameter defines the maximum number of bytes which shall be processed within one cycle of job processing.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00572] Definition of EcucIntegerParamDef NvMCsmRetryCounter

Parameter Name	NvMCsmRetryCounter
Parent Container	NvMCommon
Upstream mapping defined	–
Description	This value specifies the number of CSM encryption/decryption job retry attempts. CSM jobs for block reading and writing may fail (e.g. module busy, queue full, ...). To not directly abort the read/write with an error status, the NvM will retry the CSM job for the configured NvMCsmRetryCounter times. Configuring 0 means: no retry behavior; job will be aborted directly. **Tags:** atp.Status=draft

▽

△

Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	0	
Pre-Compile only	true	

[ECUC_NvM_00494] Definition of EcucIntegerParamDef NvMDatasetSelectionBits

Parameter Name	NvMDatasetSelectionBits	
Parent Container	NvMCommon	
Upstream mapping defined	–	
Description	Defines the number of least significant bits which shall be used to address a certain dataset of a NVRAM block within the interface to the memory hardware abstraction. 0..8: Number of bits which are used for dataset or redundant block addressing. 0: No dataset or redundant NVRAM blocks are configured at all, no selection bits required. 1: In case of redundant NVRAM blocks are configured, but no dataset NVRAM blocks. Dependency: MemHwA, NVM_NVRAM_BLOCK_IDENTIFIER, NVM_BLOCK_MANAGEMENT_TYPE	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 8	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00495] Definition of EcucBooleanParamDef NvMDevErrorDetect

Parameter Name	NvMDevErrorDetect
Parent Container	NvMCommon
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_NvM_00497] Definition of EcucBooleanParamDef NvMDynamicConfiguration

Parameter Name	NvMDynamicConfiguration
Parent Container	NvMCommon
Upstream mapping defined	–

▽

△

Description	Preprocessor switch to enable the dynamic configuration management handling by the NvM_ReadAll request. true: Dynamic configuration management handling enabled. false: Dynamic configuration management handling disabled. This parameter affects all NvM processing related to Block with ID 1 and all processing related to Resistant to Changed Software. If the Dynamic Configuration is disabled, Block 1 cannot be used by NvM.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00498] Definition of EcucBooleanParamDef NvMJobPrioritization

Parameter Name	NvMJobPrioritization
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Preprocessor switch to enable job prioritization handling true: Job prioritization handling enabled. false: Job prioritization handling disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00555] Definition of EcucFloatParamDef NvMMainFunctionPeriod

Parameter Name	NvMMainFunctionPeriod	
Parent Container	NvMCommon	
Upstream mapping defined	–	
Description	The period between successive calls to the main function in seconds.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

[ECUC_NvM_00500] Definition of EcucFunctionNameDef NvMMultiBlockCallback

Parameter Name	NvMMultiBlockCallback
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Entry address of the common callback routine which shall be invoked on termination of each asynchronous multi block request
Multiplicity	0..1
Type	EcucFunctionNameDef

▽

△

Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_NvM_00501] Definition of EcucBooleanParamDef NvMPollingMode

Parameter Name	NvMPollingMode
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Preprocessor switch to enable/disable the polling mode in the NVRAM Manager and at the same time disable/enable the callback functions useable by lower layers true: Polling mode enabled, callback function usage disabled. false: Polling mode disabled, callback function usage enabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00518] Definition of EcucIntegerParamDef NvMRepeatMirrorOperations

Parameter Name	NvMRepeatMirrorOperations	
Parent Container	NvMCommon	
Upstream mapping defined	–	
Description	Defines the number of retries to let the application copy data to or from the NvM module's mirror before postponing the current job.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	0	
Pre-Compile only	true	

[ECUC_NvM_00502] Definition of EcucBooleanParamDef NvMSetRamBlockStatusApi

Parameter Name	NvMSetRamBlockStatusApi
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Preprocessor switch to enable the API NvM_SetRamBlockStatus. true: API NvM_SetRamBlockStatus enabled. false: API NvM_SetRamBlockStatus disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_NvM_00503] Definition of EcucIntegerParamDef NvMSizeImmediateJobQueue

Parameter Name	NvMSizeImmediateJobQueue
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Defines the number of queue entries for the immediate priority job queue. If NVM_JOB_PRIORITIZATION is switched OFF this parameter shall be out of scope. Dependency: NVM_JOB_PRIORITIZATION
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	1 .. 65535
Default value	–
Pre-Compile only	true

[ECUC_NvM_00504] Definition of EcucIntegerParamDef NvMSizeStandardJobQueue

Parameter Name	NvMSizeStandardJobQueue
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Defines the number of queue entries for the standard job queue.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 65535
Default value	–
Pre-Compile only	true

[ECUC_NvM_00573] Definition of EcucChoiceReferenceDef NvMBufferAlignmentValue

Parameter Name	NvMBufferAlignmentValue
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Parameter determines the alignment of the start address that NvM buffers need to have. Value shall be inherited from EaBufferAlignmentValue or FeeBufferAlignmentValue. **Tags:** atp.Status=draft
Multiplicity	1
Type	Choice reference to [EaGeneral, FeeGeneral]
Pre-Compile only	true

[ECUC_NvM_00565] Definition of EcucReferenceDef NvMEcucPartitionRef

Parameter Name	NvMEcucPartitionRef
Parent Container	NvMCommon
Upstream mapping defined	–

▽

△

Description	Maps the NvM to one or multiple ECUC partitions to make its C-APIs available in the according partition.
Multiplicity	1..*
Type	Reference to EcucPartition
Pre-Compile only	true

[ECUC_NvM_00566] Definition of EcucReferenceDef NvMMasterEcucPartition Ref

Parameter Name	NvMMasterEcucPartitionRef
Parent Container	NvMCommon
Upstream mapping defined	–
Description	Maps the NvM master to zero or one ECUC partition to assign the master functionality to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the NvM is mapped to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

C.5 Dcm

C.5.1 DcmDspDid

[ECUC_Dcm_00601] Definition of EcucParamConfContainerDef DcmDspDid

Container Name	DcmDspDid
Parent Container	DcmDsp
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	This container contains the configuration (parameters) of the DID.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDidIdentifier	1	[ECUC_Dcm_00602]
DcmDspDidSize	0..1	[ECUC_Dcm_01099]
DcmDspDidUsed	1	[ECUC_Dcm_00805]
DcmDspDidUsePort	1	[ECUC_Dcm_01122]
DcmDspDidBndMBlockIdRef	0..1	[ECUC_Dcm_01185]
DcmDspDidInfoRef	1	[ECUC_Dcm_00604]
DcmDspDidRef	0..*	[ECUC_Dcm_00606]

Included Containers		
Container Name	**Multiplicity**	**Description**
DcmDspDidSignal	0..*	This container defines the reference to 1 DcmDspData container and position relevant for this DID.
DcmDspDidSignalCompositePool	0..*	Defines a collection of all contained signals for this root signal.
DcmDspDidSupportInfo	0..1	This container defines the support information to declare the usability of the data bytes within the DIDs

[ECUC_Dcm_00602] Definition of EcucIntegerParamDef DcmDspDidIdentifier

Parameter Name	DcmDspDidIdentifier	
Parent Container	DcmDspDid	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	2 byte Identifier of the DID Within each DcmConfigSet all DcmDspDidIdentifier values shall be unique.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_01099] Definition of EcucIntegerParamDef DcmDspDidSize

Parameter Name	DcmDspDidSize	
Parent Container	DcmDspDid	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	Length of a DID in byte(s).	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_00805] Definition of EcucBooleanParamDef DcmDspDidUsed

Parameter Name	DcmDspDidUsed	
Parent Container	DcmDspDid	
Upstream mapping defined	–	
Description	Allow to activate or deactivate the usage of a DID, for multi purpose ECUs true = DID available false = DID not available	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

[ECUC_Dcm_01122] Definition of EcucEnumerationParamDef DcmDspDidUsePort

Parameter Name	DcmDspDidUsePort	
Parent Container	DcmDspDid	
Upstream mapping defined	–	
Description	Selects application interface type for DID data elements between a single operation for all data elements or data element specific operations.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	USE_ATOMIC_BNDM	The DID shall be read/written from/to BndM.
	USE_ATOMIC_NV_DATA_INTERFACE	A single sender receiver interface with Nv Data-Ports is used for all data elements of this DID.
	USE_ATOMIC_SENDER_RECEIVER_INTERFACE	A single sender receiver interface is used to access all data elements of this DID. The sender receiver interface is typed with IsService=false.
	USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE	A single sender receiver interface is used to access all data elements of this DID. The sender receiver interface is typed with IsService=true.
	USE_DATA_ELEMENT_SPECIFIC_INTERFACES	The data elements of this DID are collected by using the data element specific interfaces defined by DcmDspDataUsePort.
Default value	USE_DATA_ELEMENT_SPECIFIC_INTERFACES	
Pre-Compile only	true	

[ECUC_Dcm_01185] Definition of EcucReferenceDef DcmDspDidBndMBlockIdRef

Parameter Name	DcmDspDidBndMBlockIdRef
Parent Container	DcmDspDid
Upstream mapping defined	–
Description	Associate this DID with a BndM blockId. Dependency: DcmDspDidUsePort == USE_ATOMIC_BNDM
Multiplicity	0..1
Type	Symbolic name reference to BndMBlockDescriptor
Pre-Compile only	true

[ECUC_Dcm_00604] Definition of EcucReferenceDef DcmDspDidInfoRef

Parameter Name	DcmDspDidInfoRef
Parent Container	DcmDspDid
Upstream mapping defined	–
Description	Reference to DcmDspDidInfo containing information on this DID.
Multiplicity	1
Type	Reference to DcmDspDidInfo
Pre-Compile only	true

[ECUC_Dcm_00606] Definition of EcucReferenceDef DcmDspDidRef

Parameter Name	DcmDspDidRef
Parent Container	DcmDspDid
Upstream mapping defined	–
Description	Reference to DcmDspDid in case this DID refer to one or serveral other DID's **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	Reference to DcmDspDid
Pre-Compile only	true

C.5.2 DcmDspDidInfo

[ECUC_Dcm_00607] Definition of EcucParamConfContainerDef DcmDspDidInfo

Container Name	DcmDspDidInfo
Parent Container	DcmDsp
Upstream mapping defined	–
Description	This container contains the configuration (parameters) of the DID's Info
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDDDIDMaxElements	0..1	[ECUC_Dcm_00970]
DcmDspDidDynamicallyDefined	1	[ECUC_Dcm_00612]

Included Containers		
Container Name	**Multiplicity**	**Description**
DcmDspDidControl	0..1	This container contains the configuration (parameters) of the DID control.
DcmDspDidRead	0..1	This container contains the configuration (parameters) of the DID read.
DcmDspDidWrite	0..1	This container contains the configuration (parameters) of the DID write.

[ECUC_Dcm_00970] Definition of EcucIntegerParamDef DcmDspDDDIDMaxElements

Parameter Name	DcmDspDDDIDMaxElements
Parent Container	DcmDspDidInfo
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Maximum number of source elements of a DDDID.
Multiplicity	0..1
Type	EcucIntegerParamDef

▽

△

Range	1 .. 255	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_00612] Definition of EcucBooleanParamDef DcmDspDidDynamicallyDefined

Parameter Name	DcmDspDidDynamicallyDefined
Parent Container	DcmDspDidInfo
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Indicates if this DID can be dynamically defined true = DID can be dynamically defined false = DID can not be dynamically defined
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

C.5.3 DcmDspData

[ECUC_Dcm_00869] Definition of EcucParamConfContainerDef DcmDspData

Container Name	DcmDspData
Parent Container	DcmDsp
Upstream mapping defined	–
Description	This container contains the configuration (parameters) of a Data belonging to a DID
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDataByteSize	0..1	[ECUC_Dcm_01106]
DcmDspDataConditionCheckReadFnc	0..1	[ECUC_Dcm_00677]
DcmDspDataConditionCheckReadFncUsed	0..1	[ECUC_Dcm_00955]
DcmDspDataEcuSignal	0..1	[ECUC_Dcm_00825]
DcmDspDataEndianness	0..1	[ECUC_Dcm_00986]
DcmDspDataFreezeCurrentStateFnc	0..1	[ECUC_Dcm_00674]
DcmDspDataGetScalingInfoFnc	0..1	[ECUC_Dcm_00676]
DcmDspDataReadDataLengthFnc	0..1	[ECUC_Dcm_00671]
DcmDspDataReadEcuSignal	0..1	[ECUC_Dcm_00824]
DcmDspDataReadFnc	0..1	[ECUC_Dcm_00669]
DcmDspDataResetToDefaultFnc	0..1	[ECUC_Dcm_00673]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDataReturnControlToEcuFnc	0..1	[ECUC_Dcm_00672]
DcmDspDataShortTermAdjustmentFnc	0..1	[ECUC_Dcm_00675]
DcmDspDataType	1	[ECUC_Dcm_00985]
DcmDspDataUsePort	1	[ECUC_Dcm_00713]
DcmDspDataWriteFnc	0..1	[ECUC_Dcm_00670]
DcmDspOdxDataDescription	0..1	[ECUC_Dcm_00988]
DcmDspDataInfoRef	0..1	[ECUC_Dcm_00811]

Included Containers		
Container Name	**Multiplicity**	**Description**
DcmDspDiagnosisScaling	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
DcmDspDidDataSupportInfo	0..1	This container defines the supported information.
DcmDspExternalSRDataElement Class	0..1	This container defines the source of data in a provided port which shall be read respectively the target of data in a required port which shall be written. This container shall contain either one DcmSubElementInData ElementInstance OR DcmDataElementInstance reference.

[ECUC_Dcm_01106] Definition of EcucIntegerParamDef DcmDspDataByteSize

Parameter Name	DcmDspDataByteSize	
Parent Container	DcmDspData, DcmDspDidSignal	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	Defines the array length in bytes or the maximum array length for variable datalengths.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_00677] Definition of EcucFunctionNameDef DcmDspDataCondition CheckReadFnc

Parameter Name	DcmDspDataConditionCheckReadFnc
Parent Container	DcmDspData
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name to demand application if the conditions (e.g. System state) to read the DID are correct. (ConditionCheckRead-function). Multiplicity shall be equal to parameter DcmDspDataReadFnc. This parameter is related to the interface Xxx_ConditionCheckRead.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–

▽

△

Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00955] Definition of EcucBooleanParamDef DcmDspDataConditionCheckReadFncUsed

Parameter Name	DcmDspDataConditionCheckReadFncUsed
Parent Container	DcmDspData
Upstream mapping defined	–
Description	This parameter determines if a condition check function is available or not.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_Dcm_00825] Definition of EcucFunctionNameDef DcmDspDataEcuSignal

Parameter Name	DcmDspDataEcuSignal
Parent Container	DcmDspData
Upstream mapping defined	–
Description	Function name to control the access to a certain ECU Signal by the DCM. (IoHwAb_Dcm_<symbolic name of ECU signal>-function).
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00986] Definition of EcucEnumerationParamDef DcmDspDataEndianness

Parameter Name	DcmDspDataEndianness	
Parent Container	DcmDspData	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	Defines the endianness of the data belonging to a DID in a diagnostic request or response message.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address.
	OPAQUE	Opaque data endianness
Pre-Compile only	true	

[ECUC_Dcm_00674] Definition of EcucFunctionNameDef DcmDspDataFreezeCurrentStateFnc

Parameter Name	DcmDspDataFreezeCurrentStateFnc
Parent Container	DcmDspData
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate], [23, CP_TPS_DiagnosticExtract Template]
Description	Function name to request to application to freeze the current state of an IOControl. (FreezeCurrentState-function). This parameter is related to the interface Xxx_FreezeCurrentState.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00676] Definition of EcucFunctionNameDef DcmDspDataGetScalingInfoFnc

Parameter Name	DcmDspDataGetScalingInfoFnc
Parent Container	DcmDspData
Upstream mapping defined	–
Description	Function name to request to application the scaling information of the DID. (GetScaling Information-function). This parameter is related to the interface Xxxx_GetScaling Information.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00671] Definition of EcucFunctionNameDef DcmDspDataReadDataLengthFnc

Parameter Name	DcmDspDataReadDataLengthFnc
Parent Container	DcmDspData
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name to request from application the data length of a DID. (ReadData Length-function). This parameter is related to the interface Xxx_ReadDataLength.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00824] Definition of EcucFunctionNameDef DcmDspDataReadEcuSignal

Parameter Name	DcmDspDataReadEcuSignal
Parent Container	DcmDspData
Upstream mapping defined	–
Description	Function name for read access to a certain ECU Signal by the DCM. (IoHwAb_Dcm_Read<EcuSignalName>-function). Only relevant if DcmDspDataUsePort==USE_ECU_SIGNAL.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00669] Definition of EcucFunctionNameDef DcmDspDataReadFnc

Parameter Name	DcmDspDataReadFnc
Parent Container	DcmDspData
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name to request from application the data value of a DID. (Read Data-function). This parameter is related to the interface Xxx_ReadData.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00673] Definition of EcucFunctionNameDef DcmDspDataResetToDefaultFnc

Parameter Name	DcmDspDataResetToDefaultFnc
Parent Container	DcmDspData
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate], [23, CP_TPS_DiagnosticExtract Template]
Description	Function name to request to application to reset an IOControl to default value. (Reset ToDefault-function). This parameter is related to the interface Xxx_ResetToDefault.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00672] Definition of EcucFunctionNameDef DcmDspDataReturnControlToEcuFnc

Parameter Name	DcmDspDataReturnControlToEcuFnc
Parent Container	DcmDspData
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name to request to application to return control to ECU of an IOControl. (ReturnControlToECU-function). This parameter is related to the interface Xxx_ReturnControlToECU.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00675] Definition of EcucFunctionNameDef DcmDspDataShortTermAdjustmentFnc

Parameter Name	DcmDspDataShortTermAdjustmentFnc
Parent Container	DcmDspData
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate], [23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name to request to application to adjust the IO signal. (ShortTermAdjustment-function). This parameter is related to the interface Xxx_ShortTermAdjustment.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00985] Definition of EcucEnumerationParamDef DcmDspDataType

Parameter Name	DcmDspDataType	
Parent Container	DcmDspData	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	Provide the implementation data type of data belonging to a DID.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BOOLEAN	Type of the data is boolean.
	DOUBLE	Type of the data is double (64 bit float).
	DOUBLE_N	Type of the data is double (64 bit float) array.
	FLOAT	Type of the data is float.
	FLOAT_N	Type of the data is float array.
	SINT16	Type of the data is sint16.
	SINT16_N	Type of the data is sint16 array.
	SINT32	Type of the data is sint32.

▽

△

	SINT32_N	Type of the data is sint32 array.
	SINT64	Type of the data is sint64.
	SINT64_N	Type of the data is sint64 array.
	SINT8	Type of the data is sint8.
	SINT8_N	Type of the data is sint8 array.
	UINT16	Type of the data is uint16.
	UINT16_N	Type of the data is uint16 array.
	UINT32	Type of the data is uint32.
	UINT32_N	Type of the data is uint32 array.
	UINT64	Type of the data is uint64.
	UINT64_N	Type of the data is uint64 array.
	UINT8	Type of the data is uint8.
	UINT8_DYN	Type of the data is uint8 array with dynamic length.
	UINT8_N	Type of the data is uint8 array.
Pre-Compile only	true	

[ECUC_Dcm_00713] Definition of EcucEnumerationParamDef DcmDspDataUsePort

Parameter Name	DcmDspDataUsePort	
Parent Container	DcmDspData	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [23, CP_TPS_DiagnosticExtract Template]	
Description	Defines which interface shall be used to access the data.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	USE_DATA_ASYNCH_CLIENT_SERVER	The DCM will access the Data using an R-Port requiring a asynchronous ClientServerInterface DataServices_{Data}. The R-Port is named Data Services_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_ASYNCH_CLIENT_SERVER_ERROR	The Dcm will access the Data using an R-Port requiring a asynchronous ClientServerInterface DataServices_{Data}. The parameter ErrorCode can be returned to allow the application to trigger a negative response during the operation. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_ASYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDsp DataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter.

▽

△

	USE_DATA_ASYNCH_FNC_ ERROR	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDsp DataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter. The parameter ErrorCode can be returned to allow the application to trigger a negative response during the operation.
	USE_DATA_ASYNCH_FNC_ PROXY	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (with DataLength and ErrorCode parameter). DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter.
	USE_DATA_SENDER_ RECEIVER	The DCM will access the Data using an Port requiring a SenderReceiverInterface (with is Service=false) DataServices_{Data}. The Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_SENDER_ RECEIVER_AS_SERVICE	The DCM will access the Data using an service Port requiring a SenderReceiverInterface (with is Service=true) DataServices_{Data} . The Port is namedDataServices_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_SYNCH_CLIENT_ SERVER	The DCM will access the Data using an R-Port requiring a synchronous ClientServerInterface DataServices_{Data}. The R-Port is named Data Services_{Data} where {Data} is the name of the container DcmDspData.
	USE_DATA_SYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDsp DataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return value is not allowed and OpStatus parameter is not existing in the prototype.
	USE_DATA_SYNCH_FNC_ PROXY	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (with DataLength and ErrorCode parameter). DCM_E_PENDING return value is not allowed and OpStatus parameter is not existing in the prototype.
	USE_ECU_SIGNAL	The DCM will access the Data using a direct access to IoHwAb
Pre-Compile only	true	

[ECUC_Dcm_00670] Definition of EcucFunctionNameDef DcmDspDataWriteFnc

Parameter Name	DcmDspDataWriteFnc
Parent Container	DcmDspData
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name to request application to write the data value of a DID. (Write Data-function). This parameter is related to the interface Xxx_WriteData.
Multiplicity	0..1
Type	EcucFunctionNameDef

▽

△

Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_00988] Definition of EcucAddInfoParamDef DcmDspOdxDataDescription

Parameter Name	DcmDspOdxDataDescription
Parent Container	DcmDspData
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Defines additional description for ODX documentation
Multiplicity	0..1
Type	EcucAddInfoParamDef
Default value	–
Pre-Compile only	true

[ECUC_Dcm_00811] Definition of EcucReferenceDef DcmDspDataInfoRef

Parameter Name	DcmDspDataInfoRef
Parent Container	DcmDspData
Upstream mapping defined	–
Description	Reference to 1 DcmDspDataInfo
Multiplicity	0..1
Type	Reference to DcmDspDataInfo
Pre-Compile only	true

C.5.4 DcmDspDataInfo

[ECUC_Dcm_00810] Definition of EcucParamConfContainerDef DcmDspDataInfo

Container Name	DcmDspDataInfo
Parent Container	DcmDsp
Upstream mapping defined	–
Description	This container contains the configuration (parameters) of one Data.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDataScalingInfoSize	0..1	[ECUC_Dcm_00611]

No Included Containers

[ECUC_Dcm_00611] Definition of EcucIntegerParamDef DcmDspDataScalingInfoSize

Parameter Name	DcmDspDataScalingInfoSize
Parent Container	DcmDspDataInfo
Upstream mapping defined	–
Description	If Scaling information service is available for this Data, it provides the size in bytes of the scaling information.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	0 .. 4294967295
Default value	–
Pre-Compile only	true

C.5.5 DcmDspPidService01

[ECUC_Dcm_00894] Definition of EcucParamConfContainerDef DcmDspPidService01

Container Name	DcmDspPidService01
Parent Container	DcmDspPidData
Upstream mapping defined	–
Description	Contains specific configuration parameter of PID for service $01. This container exists only if DcmDspPidService is set to DCM_SERVICE_01 or DCM_SERVICE_01_02.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspPidDataEndianness	0..1	[ECUC_Dcm_01012]
DcmDspPidDataReadFnc	0..1	[ECUC_Dcm_00629]
DcmDspPidDataType	1	[ECUC_Dcm_01018]
DcmDspPidDataUsePort	1	[ECUC_Dcm_00720]

Included Containers		
Container Name	**Multiplicity**	**Description**
DcmDspDiagnosisScaling	0..1	This container contains the configuration (parameters) of an alternative Diagnosis Representation. Out if this the scaling between Diagnosis and ECU internal representation and vice versa can be calculated.
DcmDspPidService01External SRDataElementClass	0..1	This container defines the source of data in a provided port which shall be read respectively the target of data in a required port which shall be written. This container shall contain either one DcmSubElementInData ElementInstance OR DcmDataElementInstance reference.

[ECUC_Dcm_01012] Definition of EcucEnumerationParamDef DcmDspPidDataEndianness

Parameter Name	DcmDspPidDataEndianness	
Parent Container	DcmDspPidService01	
Upstream mapping defined	–	
Description	Defines the endianness of the data belonging to a PID in a diagnostic response message. If no DcmDspPidDataEndianness is defined the value of DcmDspDataDefault Endianness is applicable.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address.
	OPAQUE	Opaque data endianness
Pre-Compile only	true	

[ECUC_Dcm_00629] Definition of EcucFunctionNameDef DcmDspPidDataReadFnc

Parameter Name	DcmDspPidDataReadFnc
Parent Container	DcmDspPidService01
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Function name for reading PID data value. This is only relevant if DcmDspPidDataUse Port==USE_DATA_SYNCH_FNC. This parameter is related to the interface Xxx_ReadData.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Dcm_01018] Definition of EcucEnumerationParamDef DcmDspPidDataType

Parameter Name	DcmDspPidDataType	
Parent Container	DcmDspPidService01	
Upstream mapping defined	–	
Description	Provide the implementation data type of data belonging to a PID.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BOOLEAN	Type of the data is boolean.
	DOUBLE	Type of the data is double (64 bit float).
	DOUBLE_N	Type of the data is double (64 bit float) array.
	SINT16	Type of the data is sint16.
	SINT16_N	Type of the data is sint16 array.

▽

△

	SINT32	Type of the data is sint32.
	SINT32_N	Type of the data is sint32 array.
	SINT64	Type of the data is sint64.
	SINT64_N	Type of the data is sint64 array.
	SINT8	Type of the data is sint8.
	SINT8_N	Type of the data is sint8 array.
	UINT16	Type of the data is uint16.
	UINT16_N	Type of the data is uint16 array.
	UINT32	Type of the data is uint32.
	UINT32_N	Type of the data is uint32 array.
	UINT64	Type of the data is uint64.
	UINT64_N	Type of the data is uint64 array.
	UINT8	Type of the data is uint8.
	UINT8_N	Type of the data is uint8 array.
Pre-Compile only	true	

[ECUC_Dcm_00720] Definition of EcucEnumerationParamDef DcmDspPidDataUsePort

Parameter Name	DcmDspPidDataUsePort	
Parent Container	DcmDspPidService01	
Upstream mapping defined	–	
Description	If this parameter is set to USE_DATA_SYNCH_FNC, the Dcm will use the function defined in DcmDspPidDataReadFnc to get the PID data value. If this parameter is set to USE_DATA_SYNCH_CLIENT_SERVER, the Dcm will have an R-Port requiring the interface DataServices_{Data}. If this parameter is set to USE_DATA_SENDER_RECEIVER or USE_DATA_SENDER_RECEIVER_AS_SERVICE, the DCM will have an R-Port requiring a SenderReceiver Interface. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspPidData.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	USE_DATA_SENDER_RECEIVER	The DCM will access the Data using an Port requiring a SenderReceiverInterface.
	USE_DATA_SENDER_RECEIVER_AS_SERVICE	The DCM will access the Data using an service Port requiring a SenderReceiverInterface (with is Service=true).
	USE_DATA_SYNCH_CLIENT_SERVER	The DCM will access the Data using an R-Port requiring a synchronous ClientServerInterface.
	USE_DATA_SYNCH_FNC	The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDsp DataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return value is not allowed and OpStatus parameter is not existing in the prototype.
Pre-Compile only	true	

C.5.6 DcmDspDidRead

[ECUC_Dcm_00613] Definition of EcucParamConfContainerDef DcmDspDidRead

Container Name	DcmDspDidRead
Parent Container	DcmDspDidInfo
Upstream mapping defined	–
Description	This container contains the configuration (parameters) of the DID read.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDidReadModeRuleRef	0..1	[ECUC_Dcm_00917]
DcmDspDidReadRoleRef	0..32	[ECUC_Dcm_01141]
DcmDspDidReadSecurityLevelRef	0..*	[ECUC_Dcm_00614]
DcmDspDidReadSessionRef	0..*	[ECUC_Dcm_00615]

No Included Containers

[ECUC_Dcm_00917] Definition of EcucReferenceDef DcmDspDidReadModeRuleRef

Parameter Name	DcmDspDidReadModeRuleRef
Parent Container	DcmDspDidRead
Upstream mapping defined	–
Description	Reference to DcmModeRule Mode rule which controls to read this DID. If there is no reference, no check of the mode rule shall be done.
Multiplicity	0..1
Type	Reference to DcmModeRule
Pre-Compile only	true

[ECUC_Dcm_01141] Definition of EcucReferenceDef DcmDspDidReadRoleRef

Parameter Name	DcmDspDidReadRoleRef
Parent Container	DcmDspDidRead
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspAuthenticationRow that defines a role in that this DID can be read.
Multiplicity	0..32
Type	Reference to DcmDspAuthenticationRow
Pre-Compile only	true

[ECUC_Dcm_00614] Definition of EcucReferenceDef DcmDspDidReadSecurityLevelRef

Parameter Name	DcmDspDidReadSecurityLevelRef
Parent Container	DcmDspDidRead
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspSecurityRow Referenced security levels are allowed to read this DID. If there is no reference, no check of security level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSecurityRow
Pre-Compile only	true

[ECUC_Dcm_00615] Definition of EcucReferenceDef DcmDspDidReadSessionRef

Parameter Name	DcmDspDidReadSessionRef
Parent Container	DcmDspDidRead
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspSessionRow Referenced sessions are allowed to read this DID. If there is no reference, no check of session level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSessionRow
Pre-Compile only	true

C.5.7 DcmDspDidWrite

[ECUC_Dcm_00616] Definition of EcucParamConfContainerDef DcmDspDidWrite

Container Name	DcmDspDidWrite
Parent Container	DcmDspDidInfo
Upstream mapping defined	–
Description	This container contains the configuration (parameters) of the DID write.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDidWriteModeRuleRef	0..1	[ECUC_Dcm_00922]
DcmDspDidWriteRoleRef	0..32	[ECUC_Dcm_01142]
DcmDspDidWriteSecurityLevelRef	0..*	[ECUC_Dcm_00617]
DcmDspDidWriteSessionRef	0..*	[ECUC_Dcm_00618]

No Included Containers

[ECUC_Dcm_00922] Definition of EcucReferenceDef DcmDspDidWriteModeRuleRef

Parameter Name	DcmDspDidWriteModeRuleRef
Parent Container	DcmDspDidWrite
Upstream mapping defined	–
Description	Reference to DcmModeRule Mode rule which controls to write this DID. If there is no reference, no check of the mode rule shall be done.
Multiplicity	0..1
Type	Reference to DcmModeRule
Pre-Compile only	true

[ECUC_Dcm_01142] Definition of EcucReferenceDef DcmDspDidWriteRoleRef

Parameter Name	DcmDspDidWriteRoleRef
Parent Container	DcmDspDidWrite
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspAuthenticationRow that defines a role in that this DID can be written.
Multiplicity	0..32
Type	Reference to DcmDspAuthenticationRow
Pre-Compile only	true

[ECUC_Dcm_00617] Definition of EcucReferenceDef DcmDspDidWriteSecurityLevelRef

Parameter Name	DcmDspDidWriteSecurityLevelRef
Parent Container	DcmDspDidWrite
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspSecurityRow Referenced security levels are allowed to wirte this DID. If there is no reference, no check of security level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSecurityRow
Pre-Compile only	true

[ECUC_Dcm_00618] Definition of EcucReferenceDef DcmDspDidWriteSessionRef

Parameter Name	DcmDspDidWriteSessionRef
Parent Container	DcmDspDidWrite
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspSessionRow Referenced sessions are allowed to write this DID. If there is no reference, no check of session level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSessionRow

▽

△

Pre-Compile only	true

C.5.8 DcmDspDidControl

[ECUC_Dcm_00619] Definition of EcucParamConfContainerDef DcmDspDidControl

Container Name	DcmDspDidControl
Parent Container	DcmDspDidInfo
Upstream mapping defined	–
Description	This container contains the configuration (parameters) of the DID control.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspDidControlMask	1	[ECUC_Dcm_01059]
DcmDspDidControlMaskSize	0..1	[ECUC_Dcm_01060]
DcmDspDidFreezeCurrentState	1	[ECUC_Dcm_00624]
DcmDspDidResetToDefault	1	[ECUC_Dcm_00623]
DcmDspDidShortTermAdjustment	1	[ECUC_Dcm_00625]
DcmDspDidControlModeRuleRef	0..1	[ECUC_Dcm_00923]
DcmDspDidControlRoleRef	0..32	[ECUC_Dcm_01143]
DcmDspDidControlSecurityLevelRef	0..*	[ECUC_Dcm_00620]
DcmDspDidControlSessionRef	0..*	[ECUC_Dcm_00621]

Included Containers		
Container Name	**Multiplicity**	**Description**
DcmDspDidControlEnableMask	0..32	The shortname of the container value defines the symbol of the controlMask.

[ECUC_Dcm_01059] Definition of EcucEnumerationParamDef DcmDspDidControlMask

Parameter Name	DcmDspDidControlMask
Parent Container	DcmDspDidControl
Upstream mapping defined	–
Description	This indicates the presence of "controlEnableMask" in SWC service interfaces and defines how the Dcm treats a service request.
Multiplicity	1
Type	EcucEnumerationParamDef

▽

△

Range	DCM_CONTROLMASK_EXTERNAL	The control enable mask record shall be forwarded within each interface and is handled externally.
	DCM_CONTROLMASK_INTERNAL	The control enable mask record is handled internally and Dcm controls only the included signals.
	DCM_CONTROLMASK_NO	No control enable mask handling.
Default value	DCM_CONTROLMASK_INTERNAL	
Pre-Compile only	true	

[ECUC_Dcm_01060] Definition of EcucIntegerParamDef DcmDspDidControlMaskSize

Parameter Name	DcmDspDidControlMaskSize	
Parent Container	DcmDspDidControl	
Upstream mapping defined	–	
Description	The value defines the size of the controlEnableMaskRecord in bytes.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967294	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_00624] Definition of EcucBooleanParamDef DcmDspDidFreezeCurrentState

Parameter Name	DcmDspDidFreezeCurrentState
Parent Container	DcmDspDidControl
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [23, CP_TPS_DiagnosticExtract Template]
Description	This indicates the presence of "FreezeCurrentState".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_Dcm_00623] Definition of EcucBooleanParamDef DcmDspDidResetToDefault

Parameter Name	DcmDspDidResetToDefault
Parent Container	DcmDspDidControl
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [23, CP_TPS_DiagnosticExtract Template]
Description	This indicates the presence of "ResetToDefault".
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	–
Pre-Compile only	true

[ECUC_Dcm_00625] Definition of EcucBooleanParamDef DcmDspDidShortTermAdjustment

Parameter Name	DcmDspDidShortTermAdjustment
Parent Container	DcmDspDidControl
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [23, CP_TPS_DiagnosticExtract Template]
Description	This indicates the presence of "ShortTermAdjustment".
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_Dcm_00923] Definition of EcucReferenceDef DcmDspDidControlModeRuleRef

Parameter Name	DcmDspDidControlModeRuleRef
Parent Container	DcmDspDidControl
Upstream mapping defined	–
Description	Reference to DcmModeRule Mode rule which controls this DID. If there is no reference, no check of the mode rule shall be done.
Multiplicity	0..1
Type	Reference to DcmModeRule
Pre-Compile only	true

[ECUC_Dcm_01143] Definition of EcucReferenceDef DcmDspDidControlRoleRef

Parameter Name	DcmDspDidControlRoleRef
Parent Container	DcmDspDidControl
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspAuthenticationRow that defines a role in that this IO can be be controlled.
Multiplicity	0..32
Type	Reference to DcmDspAuthenticationRow
Pre-Compile only	true

[ECUC_Dcm_00620] Definition of EcucReferenceDef DcmDspDidControlSecurityLevelRef

Parameter Name	DcmDspDidControlSecurityLevelRef
Parent Container	DcmDspDidControl
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspSecurityRow Security levels allowed to control this DID. If there is no reference, no check of security level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSecurityRow
Pre-Compile only	true

[ECUC_Dcm_00621] Definition of EcucReferenceDef DcmDspDidControlSessionRef

Parameter Name	DcmDspDidControlSessionRef
Parent Container	DcmDspDidControl
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	Reference to DcmDspSessionRow Sessions allowed to control this DID. If there is no reference, no check of session level shall be done.
Multiplicity	0..*
Type	Reference to DcmDspSessionRow
Pre-Compile only	true

C.5.9 DcmDspRoutine

[ECUC_Dcm_00640] Definition of EcucParamConfContainerDef DcmDspRoutine

Container Name	DcmDspRoutine
Parent Container	DcmDsp
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	This container contains the configuration (parameters) for Routines
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DcmDspRoutineFncSignature	1	[ECUC_Dcm_01215]
DcmDspRoutineIdentifier	1	[ECUC_Dcm_00641]
DcmDspRoutineInfoByte	0..1	[ECUC_Dcm_01063]
DcmDspRoutineUsed	1	[ECUC_Dcm_00807]
DcmDspRoutineUsePort	1	[ECUC_Dcm_00724]

Included Containers		
Container Name	**Multiplicity**	**Description**
DcmDspRequestRoutineResults	0..1	Provides the configuration of RequestResult subservice for RoutineControl service. Existence indicates that the Request RoutineResults in the RoutineControl is supported.
DcmDspStartRoutine	0..1	Provides the configuration of Start subservice for RoutineControl service.
DcmDspStopRoutine	0..1	Provides the configuration of Stop subservice for RoutineControl service. Existence indicates that the StopRoutine in the Routine Control is supported.

[ECUC_Dcm_01215] Definition of EcucEnumerationParamDef DcmDspRoutine FncSignature

Parameter Name	DcmDspRoutineFncSignature	
Parent Container	DcmDspRoutine	
Upstream mapping defined	–	
Description	If the parameter is set to ROUTINE_FNC_NORMAL the routine control functions Dcm DspStartRoutineFnc, DcmDspStopRoutineFnc or DcmDspRequestResultsRoutineFnc are defined with the signature for a cluster local user with separate arguments per signal. If the parameter is set to ROUTINE_FNC_PROXY the routine control functions DcmDspStartRoutineFnc, DcmDspStopRoutineFnc or DcmDspRequestResultsRoutine Fnc with a fixed number of arguments passing a common buffer for all signals and a DataLength.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ROUTINE_FNC_NORMAL	Function calls with normal set of arguments used for Sw Cluster internal interfaces.
	ROUTINE_FNC_PROXY	Function call used with generic set of arguments used for Dcm low proxy.
Default value	ROUTINE_FNC_NORMAL	
Pre-Compile only	true	

[ECUC_Dcm_00641] Definition of EcucIntegerParamDef DcmDspRoutineIdentifier

Parameter Name	DcmDspRoutineIdentifier	
Parent Container	DcmDspRoutine	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	2 bytes Identifier of the RID Within each DcmConfigSet all DcmDspRoutineIdentifier values shall be unique.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_01063] Definition of EcucIntegerParamDef DcmDspRoutineInfoByte

Parameter Name	DcmDspRoutineInfoByte	
Parent Container	DcmDspRoutine	
Upstream mapping defined	–	
Description	If configured it defines the static value used by the Dcm to report as routine info byte. An example of a possible usage are routines within the OBD range of RID 0xE000-0x E1FF. If the parameter is not configured, no Dcm created routine info byte is provided within the positive response of a RoutineControl Service.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

[ECUC_Dcm_00807] Definition of EcucBooleanParamDef DcmDspRoutineUsed

Parameter Name	DcmDspRoutineUsed	
Parent Container	DcmDspRoutine	
Upstream mapping defined	–	
Description	Allow to activate or deactivate the usage of a Routine, for multi purpose ECUs True = Routine is available False = Routine is not available	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

[ECUC_Dcm_00724] Definition of EcucBooleanParamDef DcmDspRoutineUsePort

Parameter Name	DcmDspRoutineUsePort
Parent Container	DcmDspRoutine
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	If this parameter is set to true, the DCM uses a port requiring a PortInterface Routine Services_{RoutineName}. The R-Port is named RoutineServices_{RoutineName} where {RoutineName} is the name of the container DcmDspRoutine In that case, the configuration must not provide function names in DcmDspStartRoutineFnc, DcmDspStopRoutineFnc or DcmDsp RequestResultsRoutineFnc. If this is false, the DCM expects to find the names of the functions to be used in DcmDspStartRoutineFnc, DcmDspStopRoutineFnc or DcmDsp RequestResultsRoutineFnc.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

C.6 Dem

C.6.1 DemEventParameter

[ECUC_Dem_00661] Definition of EcucParamConfContainerDef DemEventParameter

Container Name	DemEventParameter
Parent Container	DemConfigSet
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate]
Description	This container contains the configuration (parameters) for events.
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DemCausalityDelayTime	0..1	[ECUC_Dem_00921]
DemComponentPriority	0..1	[ECUC_Dem_00909]
DemEventAssociatedIdentification	0..1	[ECUC_Dem_00969]
DemEventAvailable	1	[ECUC_Dem_00792]
DemEventAvailableInVariant	0..1	[ECUC_Dem_00968]
DemEventConfirmationThreshold	1	[ECUC_Dem_00924]
DemEventConfirmationThresholdCounterAdaptable	0..1	[ECUC_Dem_00929]
DemEventId	1	[ECUC_Dem_00659]
DemEventKind	1	[ECUC_Dem_00660]
DemEventRecoverableInSameOperationCycle	0..1	[ECUC_Dem_00916]
DemEventReportingType	1	[ECUC_Dem_00970]
DemFFPrestorageInNvm	0..1	[ECUC_Dem_00948]
DemFFPrestorageSupported	1	[ECUC_Dem_00671]
DemReportBehavior	0..1	[ECUC_Dem_00894]
DemComponentClassRef	0..1	[ECUC_Dem_00908]
DemDTCRef	0..1	[ECUC_Dem_00888]
DemEnableConditionGroupRef	0..1	[ECUC_Dem_00746]
DemOBDGroupingAssociativeEventsRef	0..1	[ECUC_Dem_00839]
DemOperationCycleRef	1	[ECUC_Dem_00702]
DemStorageConditionGroupRef	0..1	[ECUC_Dem_00769]

Included Containers		
Container Name	**Multiplicity**	**Description**
DemCallbackClearEventAllowed	0..1	The presence of this container indicates that the Dem has access to a "ClearEventAllowed" callback. In case there is a DemCallbackClearEventAllowedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackClearEventAllowedFnc, the Dem will have an R-Port requiring the interface CallbackClear EventAllowed whose name is generated by using the unique callback-prefix followed by the event name.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
DemCallbackEventDataChanged	0..1	The presence of this container indicates that the Dem has access to an "EventDataChanged" callback. In case there is a DemCallbackEventDataChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackEventDataChangedFnc, the Dem will have an R-Port requiring the interface CallbackEvent DataChanged whose name is generated by using the unique callback-prefix followed by the event name.
DemCallbackEventUdsStatus Changed	0..*	The presence of this container indicates, that the Dem has access to an "EventUdsStatusChanged" callback, which the Dem will call to notify other components about the change in the status of an event. In case there is a DemCallbackEventUdsStatusChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackEventUdsStatusChangedFnc, the Dem will have an R-Port requiring the interface Callback EventUdsStatusChanged, whose name is generated by using the unique callback-prefix followed by the event name.
DemCallbackInitMForE	0..1	The presence of this container indicates, that the Dem has access to an "InitMonitorForEvent" callback, which the Dem will call to initialize a monitor. In case the container has a DemCallbackInitMForEFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackInitMForEFnc, the Dem will have an R-Port requiring the interface CallbackInitMonitorForEvent, whose name is generated by using the unique callback-prefix followed by the event name.
DemCallbackMonitorStatus Changed	0..*	The presence of this container indicates, that the Dem has access to an "MonitorStatusChanged" callback, which the Dem will call to notify other components about the change in the status of an event. In case there is a DemCallbackMonitorStatusChangedFnc, this parameter defines the name of the function that the Dem will call. In case there is no DemCallbackMonitorStatusChangedFnc, the Dem will have an R-Port requiring the interface CallbackMonitor StatusChanged, whose name is generated by using the unique callback-prefix followed by the event name.
DemDebounceAlgorithmClass	1	Debounce algorithm class: counter based, time based, or monitor internal.
DemIndicatorAttribute	0..255	This container contains the event specific configuration of Indicators.

[ECUC_Dem_00921] Definition of EcucFloatParamDef DemCausalityDelayTime

Parameter Name	DemCausalityDelayTime	
Parent Container	DemEventParameter	
Upstream mapping defined	–	
Description	Time to wait until the event is considered as causal. The parameter is specified in seconds.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 2.5]	

▽

△

Default value	0
Pre-Compile only	true

[ECUC_Dem_00909] Definition of EcucIntegerParamDef DemComponentPriority

Parameter Name	DemComponentPriority	
Parent Container	DemEventParameter	
Upstream mapping defined	–	
Description	Specifies the priority within the component. A lower value means higher priority.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

[ECUC_Dem_00969] Definition of EcucIntegerParamDef DemEventAssociatedIdentification

Parameter Name	DemEventAssociatedIdentification	
Parent Container	DemEventParameter	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	Event associated identifier that allows to identify an event. This value can be reported as internal data element in snapshot records or extended data records.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

[ECUC_Dem_00792] Definition of EcucBooleanParamDef DemEventAvailable

Parameter Name	DemEventAvailable	
Parent Container	DemEventParameter	
Upstream mapping defined	–	
Description	This parameter configures an Event as unavailable. It is treated by Dem as if it does not exist. true = Event is available false = Event is not available	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

[ECUC_Dem_00968] Definition of EcucBooleanParamDef DemEventAvailableIn Variant

Parameter Name	DemEventAvailableInVariant	
Parent Container	DemEventParameter	
Upstream mapping defined	–	
Description	This parameter defines the enabled or disabled status of an event. If set to FALSE, the Event is generally disabled, even though the SW-C is integrated and the Event is described in the ECUC. The Dem shall ignore all calls from the SW-C. If set to TRUE, the availability of the event depends on other availability sources.	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	true	
Post-Build Value	Loadable	Selectable

[ECUC_Dem_00924] Definition of EcucIntegerParamDef DemEventConfirmation Threshold

Parameter Name	DemEventConfirmationThreshold	
Parent Container	DemEventParameter	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	Defines the operation cycle threshold of the DTC confirmation status according "Confirmation Threshold" of ISO 14229-1.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	1	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Dem_00929] Definition of EcucBooleanParamDef DemEventConfirmationThresholdCounterAdaptable

Parameter Name	DemEventConfirmationThresholdCounterAdaptable
Parent Container	DemEventParameter
Upstream mapping defined	–
Description	Indicates whether the events confirmation Cycle threshold can be adapted by Dem_SetEventConfirmationThresholdCounter.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Dem_00659] Definition of EcucIntegerParamDef DemEventId

Parameter Name	DemEventId
Parent Container	DemEventParameter
Upstream mapping defined	–

▽

△

Description	Unique identifier of a diagnostic event. This parameter should not be changeable by user, because the Id should be generated by Dem itself to prevent gaps and multiple use of an Id. The events should be sequentially ordered beginning with 1 and no gaps in between.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

[ECUC_Dem_00660] Definition of EcucEnumerationParamDef DemEventKind

Parameter Name	DemEventKind	
Parent Container	DemEventParameter	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	This parameter is used to distinguish between SW-C and BSW events.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DEM_EVENT_KIND_BSW	The event is a assigned to a BSW module
	DEM_EVENT_KIND_SWC	The event is a assigned to a SW-C
Default value	DEM_EVENT_KIND_SWC	
Pre-Compile only	true	

[ECUC_Dem_00916] Definition of EcucBooleanParamDef DemEventRecoverable InSameOperationCycle

Parameter Name	DemEventRecoverableInSameOperationCycle	
Parent Container	DemEventParameter	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	If parameter is configured to FALSE, reporting of PASSED will be ignored if the event is already "testfailed this operation cycle".	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	true	
Post-Build Value	-	Selectable

[ECUC_Dem_00970] Definition of EcucEnumerationParamDef DemEventReportingType

Parameter Name	DemEventReportingType
Parent Container	DemEventParameter
Upstream mapping defined	–
Description	This parameter defines the way a monitor can report an event to the Dem. There are various ways to report an event and the Dem provides different APIs for it. **Tags:** atp.Status=draft
Multiplicity	1

▽

△

Type	EcucEnumerationParamDef	
Range	STANDARD_REPORTING	The event is reported over the API Dem_SetEventStatus. **Tags:** atp.Status=draft
	STANDARD_REPORTING_WITH_MONITOR_DATA	The event is reported over the API Dem_SetEventStatusWithMonitorData. **Tags:** atp.Status=draft
Default value	STANDARD_REPORTING	
Pre-Compile only	true	

[ECUC_Dem_00948] Definition of EcucBooleanParamDef DemFFPrestorageInNvm

Parameter Name	DemFFPrestorageInNvm
Parent Container	DemEventParameter
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [23, CP_TPS_DiagnosticExtractTemplate]
Description	If the event uses a pre-stored freeze-frame this attribute indicates if the event requires the pre-stored data to be stored in non-volatile memory. TRUE = store the pre-stored data in non-volatile memory, FALSE = pre-stored data is not stored in non-volatile memory.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

[ECUC_Dem_00671] Definition of EcucBooleanParamDef DemFFPrestorageSupported

Parameter Name	DemFFPrestorageSupported
Parent Container	DemEventParameter
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	If this parameter is set to true, then the Prestorage of FreezeFrames is supported by the assigned event. This parameter is useful to calculate the buffer size.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Dem_00894] Definition of EcucEnumerationParamDef DemReportBehavior

Parameter Name	DemReportBehavior
Parent Container	DemEventParameter
Upstream mapping defined	–

▽

△

Description	Indicates the reporting behavior of the BSW Module (DemEventKind == DEM_EVENT_KIND_BSW) in order to determine the size of the reporting queue. If the parameter is not defined it means REPORT_BEFORE_INIT.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	REPORT_AFTER_INIT	Indicates that the Event will not be reported before Dem_Init().
	REPORT_BEFORE_INIT	Indicates that the Event may be reported before Dem_Init().
Default value	REPORT_BEFORE_INIT	
Pre-Compile only	true	

[ECUC_Dem_00908] Definition of EcucReferenceDef DemComponentClassRef

Parameter Name	DemComponentClassRef
Parent Container	DemEventParameter
Upstream mapping defined	–
Description	Reference to the monitored component.
Multiplicity	0..1
Type	Reference to DemComponent
Pre-Compile only	true

[ECUC_Dem_00888] Definition of EcucReferenceDef DemDTCRef

Parameter Name	DemDTCRef
Parent Container	DemEventParameter
Upstream mapping defined	–
Description	This parameter defines the DTC configuration (typically Uds) associated with the diagnostic event. It is allowed to have events without a DTC (e.g. for ECU-internal events triggering safety reactions without being reported via diagnostic communication). The same Dem DTCAttributes can be used from several events, to combine these (refer to chapter "Combination of diagnostic event").
Multiplicity	0..1
Type	Reference to DemDTC
Pre-Compile only	true

[ECUC_Dem_00746] Definition of EcucReferenceDef DemEnableConditionGroupRef

Parameter Name	DemEnableConditionGroupRef	
Parent Container	DemEventParameter	
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]	
Description	References an enable condition group.	
Multiplicity	0..1	
Type	Reference to DemEnableConditionGroup	
Post-Build Multiplicity	Loadable	Selectable

▽

△

Post-Build Value	Loadable	Selectable

[ECUC_Dem_00839] Definition of EcucReferenceDef DemOBDGroupingAssociativeEventsRef

Parameter Name	DemOBDGroupingAssociativeEventsRef
Parent Container	DemEventParameter
Upstream mapping defined	–
Description	This parameter defines a reference which points to a representative event of one group of associate events. The "reverence event" must refer to it self. Note: One event is only allowed to be reverenced to only one group of associate events.
Multiplicity	0..1
Type	Reference to DemEventParameter
Pre-Compile only	true

[ECUC_Dem_00702] Definition of EcucReferenceDef DemOperationCycleRef

Parameter Name	DemOperationCycleRef	
Parent Container	DemEventParameter	
Upstream mapping defined	–	
Description	Kind of operation cycle for the event (e.g. power cycle, driving cycle, ...)	
Multiplicity	1	
Type	Reference to DemOperationCycle	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Dem_00769] Definition of EcucReferenceDef DemStorageConditionGroupRef

Parameter Name	DemStorageConditionGroupRef
Parent Container	DemEventParameter
Upstream mapping defined	[23, CP_TPS_DiagnosticExtractTemplate]
Description	References a storage condition group.
Multiplicity	0..1
Type	Reference to DemStorageConditionGroup
Pre-Compile only	true

C.6.2 DemDataElementClass

[ECUC_Dem_00610] Definition of EcucChoiceContainerDef DemDataElementClass

Choice Container Name	DemDataElementClass
Parent Container	DemGeneral
Upstream mapping defined	–
Description	This container contains the configuration (parameters) for an external sender/receiver based data element class. It defines, how the Dem can obtain the value of the data element from a SW-C, by using a sender/receiver port.
Multiplicity	0..65535
Pre-Compile only	true

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Description**
DemExternalCSDataElementClass	0..1	This container contains the configuration (parameters) for an external client/server based data element class. It defines, how the Dem can obtain the value of the data element from either a SW-C or another BSW module. Whether a client/server port or a C function-call is used, is defined by DemDataElementUsePort.
DemExternalSRDataElementClass	0..1	This container contains the configuration (parameters) for an external sender/receiver based data element class. It defines, how the Dem can obtain the value of the data element from a SW-C, by using a sender/receiver port.
DemInternalDataElementClass	0..1	This container contains the configuration (parameters) for an internal data element class.

C.7 FiM

C.7.1 FiMFID

[ECUC_FiM_00039] Definition of EcucParamConfContainerDef FiMFID

Container Name	FiMFID
Parent Container	FiMConfigSet
Upstream mapping defined	[22, CP_TPS_BSWModuleDescriptionTemplate], [23, CP_TPS_DiagnosticExtract Template]
Description	This container includes symbolic names of all FIDs.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FiMFunctionId	1	[ECUC_FiM_00085]
FiMInhibitionTracking	1	[ECUC_FiM_00616]

Included Containers		
Container Name	**Multiplicity**	**Description**
FiMCallbackFIDStatusChanged	0..1	The presence of this container sets up the callout function FIDStatusChanged for this FID.

[ECUC_FiM_00085] Definition of EcucIntegerParamDef FiMFunctionId

Parameter Name	FiMFunctionId
Parent Container	FiMFID
Upstream mapping defined	–
Description	Unique identifier of a FimFunctionId. This parameter should not be changeable by user, because the Id should be generated by Fim itself to prevent gaps and multiple use of an Id. Note: The implementer can add the attribute 'withAuto' to the parameter definition which indicates that the value can be calculated by the generator automatically. When 'withAuto' is set to 'true' for this parameter definition the 'isAutoValue' can be set to 'true'. If 'isAutoValue' is set to 'true' the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
Pre-Compile only	true

[ECUC_FiM_00616] Definition of EcucBooleanParamDef FiMInhibitionTracking

Parameter Name	FiMInhibitionTracking
Parent Container	FiMFID
Upstream mapping defined	–
Description	Defines if inhibition tracking is required for this FID • True: inhibition tracking is enabled • False: inhibition tracking is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

C.8 Com

C.8.1 ComMainFunctionTx

[ECUC_Com_10014] Definition of EcucParamConfContainerDef ComMainFunctionTx

Container Name	ComMainFunctionTx
Parent Container	ComConfig
Upstream mapping defined	–
Description	Each element of this container defines one instance of Com_MainFunctionTx.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMainTxTimeBase	1	[ECUC_Com_10010]
ComPreparationNotification	0..1	[ECUC_Com_10020]
ComMainTxPartitionRef	0..1	[ECUC_Com_10019]

No Included Containers

[ECUC_Com_10010] Definition of EcucFloatParamDef ComMainTxTimeBase

Parameter Name	ComMainTxTimeBase
Parent Container	ComMainFunctionTx
Upstream mapping defined	–
Description	The period between successive calls to according instance of Com_MainFunctionTx in seconds. This parameter may be used by the COM generator to transform the values of the reception related timing configuration parameters of the COM module to internal implementation specific counter or tick values. The COM module's internal timing handling is implementation specific. The COM module (generator) may rely on the fact that Com_MainFunctionTx is scheduled according to the value configured here.
Multiplicity	1
Type	EcucFloatParamDef
Range	]0 .. INF[
Default value	–
Pre-Compile only	true

[ECUC_Com_10020] Definition of EcucFunctionNameDef ComPreparationNotification

Parameter Name	ComPreparationNotification
Parent Container	ComMainFunctionTx
Upstream mapping defined	–
Description	This callback function indicates that the signals/signal groups to be sent via a dedicated Com_MainFunctionTx instance will now be prepared for transmission. If this parameter is omitted no notification shall take place.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Com_10019] Definition of EcucReferenceDef ComMainTxPartitionRef

Parameter Name	ComMainTxPartitionRef
Parent Container	ComMainFunctionTx
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according Com_MainFunction instance is assigned to.
Multiplicity	0..1

▽

△

Type	Reference to EcucPartition
Pre-Compile only	true

C.8.2 ComMainFunctionRx

[ECUC_Com_10011] Definition of EcucParamConfContainerDef ComMainFunctionRx

Container Name	ComMainFunctionRx
Parent Container	ComConfig
Upstream mapping defined	–
Description	Each element of this container defines one instance of Com_MainFunctionRx.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMainRxTimeBase	1	[ECUC_Com_10018]
ComMainRxPartitionRef	0..1	[ECUC_Com_10017]

No Included Containers

[ECUC_Com_10018] Definition of EcucFloatParamDef ComMainRxTimeBase

Parameter Name	ComMainRxTimeBase	
Parent Container	ComMainFunctionRx	
Upstream mapping defined	–	
Description	The period between successive calls to according instance of Com_MainFunctionRx in seconds. This parameter may be used by the COM generator to transform the values of the reception related timing configuration parameters of the COM module to internal implementation specific counter or tick values. The COM module's internal timing handling is implementation specific. The COM module (generator) may rely on the fact that Com_MainFunctionRx is scheduled according to the value configured here.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

[ECUC_Com_10017] Definition of EcucReferenceDef ComMainRxPartitionRef

Parameter Name	ComMainRxPartitionRef
Parent Container	ComMainFunctionRx
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according Com_MainFunction instance is assigned to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

C.8.3 ComGroupSignal

[ECUC_Com_00520] Definition of EcucParamConfContainerDef ComGroupSignal

Container Name	ComGroupSignal	
Parent Container	ComSignalGroup	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	This container contains the configuration parameters of group signals. I.e. signals that are included within a signal group.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComBitPosition	1	[ECUC_Com_00259]
ComBitSize	0..1	[ECUC_Com_00158]
ComGroupSignalTransferProperty	0..1	[ECUC_Com_00560]
ComHandleId	0..1	[ECUC_Com_00165]
ComSignalDataInvalidValue	0..1	[ECUC_Com_00391]
ComSignalEndianness	1	[ECUC_Com_00157]
ComSignalInitValue	0..1	[ECUC_Com_00170]
ComSignalLength	0..1	[ECUC_Com_00437]
ComSignalType	1	[ECUC_Com_00127]
ComTimeoutSubstitutionValue	0..1	[ECUC_Com_10006]
ComSystemTemplateSystemSignalRef	0..1	[ECUC_Com_00002]

Included Containers		
Container Name	**Multiplicity**	**Description**
ComFilter	0..1	This container contains the configuration parameters of the AUTOSAR COM module's filters. Note: On sender side the container is used to specify the transmission mode conditions.

[ECUC_Com_00259] Definition of EcucIntegerParamDef ComBitPosition

Parameter Name	ComBitPosition	
Parent Container	ComGroupSignal, ComGwDestinationDescription, ComGwSourceDescription, ComSignal	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Starting position within the I-PDU. This parameter refers to the position in the I-PDU and not in the shadow buffer. If the endianness conversion is configured to Opaque the parameter ComBitPosition shall define the bit0 of the first byte like in little endian byte order	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

[ECUC_Com_00158] Definition of EcucIntegerParamDef ComBitSize

Parameter Name	ComBitSize	
Parent Container	ComGroupSignal, ComGwSourceDescription, ComSignal	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Size in bits, for integer signal types. For ComSignalType UINT8_N and UINT8_DYN the size shall be configured by ComSignalLength. For ComSignalTypes FLOAT32 and FLOAT64 the size is already defined by the signal type and therefore may be omitted.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 64	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_00560] Definition of EcucEnumerationParamDef ComGroupSignal TransferProperty

Parameter Name	ComGroupSignalTransferProperty	
Parent Container	ComGroupSignal	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Optionally defines whether this group signal shall contribute to the TRIGGERED_ON_CHANGE transfer property of the signal group. If at least one group signal of a signal group has the "ComGroupSignalTransferProperty" configured all other group signals of that signal group shall have the attribute configured as well.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	PENDING	A change of the value of this group signal shall not be considered in the evaluation of the signal groups ComTransferProperty.
	TRIGGERED_ON_CHANGE	A change of the value of this group signal shall be considered in the evaluation of the signal groups ComTransferProperty.
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_00165] Definition of EcucIntegerParamDef ComHandleId

Parameter Name	ComHandleId
Parent Container	ComGroupSignal, ComSignal, ComSignalGroup
Upstream mapping defined	–
Description	The numerical value used as the ID. This ID identifies signals and signal groups in the COM APIs using Com_SignalIdType or Com_SignalGroupIdType parameter respectively.
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

[ECUC_Com_00391] Definition of EcucStringParamDef ComSignalDataInvalidValue

Parameter Name	ComSignalDataInvalidValue
Parent Container	ComGroupSignal, ComSignal
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [13, CP_TPS_SystemTemplate]
Description	Defines the data invalid value of the signal. In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification. In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification. In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification. In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal. Dependency: In case of UINT8_N the length of ComSignalDataInvalidValue has to be the same as ComSignalLength.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Com_00157] Definition of EcucEnumerationParamDef ComSignalEndianness

Parameter Name	ComSignalEndianness
Parent Container	ComGroupSignal, ComGwDestinationDescription, ComGwSourceDescription, ComSignal
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Defines the endianness of the signal's network representation.
Multiplicity	1

▽

△

Type	EcucEnumerationParamDef	
Range	BIG_ENDIAN	Most significant byte shall be stored at the lowest address.
	LITTLE_ENDIAN	Most significant byte shall be stored at the highest address.
	OPAQUE	Opaque data endianness
Post-Build Value	Loadable	Selectable

[ECUC_Com_00170] Definition of EcucStringParamDef ComSignalInitValue

Parameter Name	ComSignalInitValue	
Parent Container	ComGroupSignal, ComGwDestinationDescription, ComSignal	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [13, CP_TPS_SystemTemplate]	
Description	Initial value for this signal. In case of UINT8_N the default value is a string of length ComSignalLength with all bytes set to 0x00. In case of UINT8_DYN the initial size shall be 0. In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification. In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification. In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification. In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal. Dependency: In case of UINT8_N the length of ComSignalInitValue has to be the same as ComSignalLength.	
Multiplicity	0..1	
Type	EcucStringParamDef	
Default value	0	
Regular Expression	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_00437] Definition of EcucIntegerParamDef ComSignalLength

Parameter Name	ComSignalLength
Parent Container	ComGroupSignal, ComGwSourceDescription, ComSignal
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Description: For ComSignalType UINT8_N this parameter specifies the length n in bytes. For ComSignalType UINT8_DYN it specifies the maximum length in bytes. For all other types this parameter shall be ignored. The supported maximum length is restricted by the used transportation system. For non TP-PDUs the maximum size of a PDU, and therefore also of any included signal, is limited by the concrete bus characteristic. For example, the limit is 8 bytes for CAN and LIN, 64 bytes for CAN FD and 254 for FlexRay.
Multiplicity	0..1
Type	EcucIntegerParamDef

▽

△

Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

[ECUC_Com_00127] Definition of EcucEnumerationParamDef ComSignalType

Parameter Name	ComSignalType	
Parent Container	ComGroupSignal, ComGwSourceDescription, ComSignal	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The AUTOSAR type of the signal. Whether or not the signal is signed or unsigned can be found by examining the value of this attribute. This type could also be used to reserved appropriate storage in AUTOSAR COM.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BOOLEAN	Type of the data is boolean.
	FLOAT32	Type of the data is float32.
	FLOAT64	Type of the data is float64.
	SINT16	Type of the data is sint16.
	SINT32	Type of the data is sint32.
	SINT64	Type of the data is sint64.
	SINT8	Type of the data is sint8.
	UINT16	Type of the data is uint16.
	UINT32	Type of the data is uint32.
	UINT64	Type of the data is uint64.
	UINT8	Type of the data is uint8.
	UINT8_DYN	Type of the data is uint8 array with dynamic length.
	UINT8_N	Type of the data is uint8 array.
Pre-Compile only	true	

[ECUC_Com_10006] Definition of EcucStringParamDef ComTimeoutSubstitution Value

Parameter Name	ComTimeoutSubstitutionValue
Parent Container	ComGroupSignal, ComSignal
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [13, CP_TPS_SystemTemplate]

▽

△

Description	The signal substitution value will be used in case of a timeout and ComRxDataTimeout Action is set to SUBSTITUTE. • In case of UINT8_N, the default value is a string of length ComSignalLength with all bytes set to 0x00. • In case of UINT8_DYN, the initial size shall be 0. • In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64, the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification. • In case the ComSignalType is FLOAT32, FLOAT64, the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification. • In case the ComSignalType is BOOLEAN, the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification. • In case the ComSignal is a UINT8_N, UINT8_DYN, the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN, the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.	
Multiplicity	0..1	
Type	EcucStringParamDef	
Default value	–	
Regular Expression	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_00002] Definition of EcucForeignReferenceDef ComSystemTemplateSystemSignalRef

Parameter Name	ComSystemTemplateSystemSignalRef	
Parent Container	ComGroupSignal, ComSignal	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Reference to the ISignalToIPduMapping that contains a reference to the ISignal (System Template) which this ComSignal (or ComGroupSignal) represents.	
Multiplicity	0..1	
Type	Foreign reference to I-SIGNAL-TO-I-PDU-MAPPING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

C.8.4 ComIPdu

[ECUC_Com_00340] Definition of EcucParamConfContainerDef ComIPdu

Container Name	ComIPdu
Parent Container	ComConfig
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Contains the configuration parameters of the AUTOSAR COM module's I-PDUs.

▽

△

Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComIPduCallout	0..1	[ECUC_Com_00387]
ComIPduCancellationSupport	0..1	[ECUC_Com_00709]
ComIPduDirection	1	[ECUC_Com_00493]
ComIPduHandleId	0..1	[ECUC_Com_00175]
ComIPduLargeData	1	[ECUC_Com_10045]
ComIPduSignalProcessing	1	[ECUC_Com_00119]
ComIPduTriggerTransmitCallout	0..1	[ECUC_Com_00765]
ComIPduType	1	[ECUC_Com_00761]
ComIPduGroupRef	0..*	[ECUC_Com_00206]
ComIPduMainFunctionRef	0..1	[ECUC_Com_10012]
ComIPduSignalGroupRef	0..*	[ECUC_Com_00519]
ComIPduSignalRef	0..*	[ECUC_Com_00518]
ComMainFunctionRouteSignalsRef	0..1	[ECUC_Com_10021]
ComPduIdRef	1	[ECUC_Com_00711]

Included Containers		
Container Name	**Multiplicity**	**Description**
ComMetaDataDefaultItem	0..*	Defines a default value for a meta data item. Used for sending an I-PDU with meta data when it is triggered spontaneously (and not by Com_TriggerIPDUSendWithMetaData), and no meta data has been provided by the RTE. Also used for reception, when e.g. Com_ReceiveSignalWithMetaData is called before an I-PDU is received. It represents a MetaDataItem of the referenced global PDU.
ComTxIPdu	0..1	This container must be included if COM_IPDU_DIRECTION is configured to SEND.

[ECUC_Com_00387] Definition of EcucFunctionNameDef ComIPduCallout

Parameter Name	ComIPduCallout
Parent Container	ComIPdu
Upstream mapping defined	–
Description	This parameter defines the existence and the name of a callout function for the corresponding I-PDU. If this parameter is omitted no I-PDU callout shall take place for the corresponding I-PDU.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Com_00709] Definition of EcucBooleanParamDef ComIPduCancellationSupport

Parameter Name	ComIPduCancellationSupport	
Parent Container	ComIPdu	
Upstream mapping defined	–	
Description	Defines for I-PDUs with ComIPduType NORMAL: If the underlying IF-modul supports cancellation of transmit requests. Defines for I-PDUs with ComIPduType TP: If the underlying TP-module supports RX and TX cancellation of ongoing requests. Dependency: This parameter shall not be set to true if ComCancellationSupport is set to false	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_00493] Definition of EcucEnumerationParamDef ComIPduDirection

Parameter Name	ComIPduDirection	
Parent Container	ComIPdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	The direction defines if this I-PDU, and therefore the contributing signals and signal groups, shall be sent or received. Dependency: If configured to Sent also a ComTxIpdu container shall be included, see ECUC_Com_00496	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	RECEIVE	Receive
	SEND	Send
Pre-Compile only	true	

[ECUC_Com_00175] Definition of EcucIntegerParamDef ComIPduHandleId

Parameter Name	ComIPduHandleId
Parent Container	ComIPdu
Upstream mapping defined	–
Description	The numerical value used as the ID of this I-PDU. The ComIPduHandleId is required by the API calls Com_RxIndication, Com_TpRxIndication, Com_StartOfReception and Com_CopyRxData to receive I-PDUs from the PduR (ComIP-duDirection: Receive), as well as the PduId passed to an Rx-I-PDU-callout. For Tx-I-PDUs (ComIPduDirection: Send), this handle Id is used for the APIs calls Com_TxConfirmation, Com_Trigger Transmit, Com_TriggerIPDUSend or Com_TriggerIPDUSendWithMetaData, Com_Copy TxData and Com_TpTxConfirmation to transmit respectively confirm transmissions of I-PDUs, as well as the PduId passed to the Tx-I-PDU-callout configured with ComIPdu Callout and/or ComIPduTriggerTransmitCallout. This ID is used also in Com_Transmit for I-PDUs with ComIPduLargeData set to true.
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)

▽

△

Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

[ECUC_Com_10045] Definition of EcucBooleanParamDef ComIPduLargeData

Parameter Name	ComIPduLargeData
Parent Container	ComIPdu
Upstream mapping defined	–
Description	This parameter enables/disables Large Data Path support for the I-PDU.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Com_00119] Definition of EcucEnumerationParamDef ComIPduSignalProcessing

Parameter Name	ComIPduSignalProcessing	
Parent Container	ComIPdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	For the definition of the two modes Immediate and Deferred.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DEFERRED	signal indication / confirmations are deferred for example to a cyclic task
	IMMEDIATE	the signal indications / confirmations are performed in Com_RxIndication/ Com_Tx Confirmation
Post-Build Value	Loadable	Selectable

[ECUC_Com_00765] Definition of EcucFunctionNameDef ComIPduTriggerTransmitCallout

Parameter Name	ComIPduTriggerTransmitCallout
Parent Container	ComIPdu
Upstream mapping defined	–
Description	If there is a trigger transmit callout defined for this I-PDU this parameter contains the name of the callout function.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

[ECUC_Com_00761] Definition of EcucEnumerationParamDef ComIPduType

Parameter Name	ComIPduType	
Parent Container	ComIPdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Defines if this I-PDU is a normal I-PDU that can be sent unfragmented or if this is a large I-PDU that shall be sent via the Transport Protocol of the underlying bus.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	NORMAL	sent or received via normal L-PDU
	TP	sent or received via TP
Post-Build Value	Loadable	Selectable

[ECUC_Com_00206] Definition of EcucReferenceDef ComIPduGroupRef

Parameter Name	ComIPduGroupRef	
Parent Container	ComIPdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Reference to the I-PDU groups this I-PDU belongs to.	
Multiplicity	0..*	
Type	Reference to ComIPduGroup	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_10012] Definition of EcucChoiceReferenceDef ComIPduMainFunctionRef

Parameter Name	ComIPduMainFunctionRef
Parent Container	ComIPdu
Upstream mapping defined	–
Description	Reference to the Com_MainFunctionRx/Com_MainFunctionTx this I-PDU belongs to. Mandatory, if multiple main functions of the relevant type are defined.
Multiplicity	0..1
Type	Choice reference to [ComMainFunctionRx, ComMainFunctionTx]
Pre-Compile only	true

[ECUC_Com_00519] Definition of EcucReferenceDef ComIPduSignalGroupRef

Parameter Name	ComIPduSignalGroupRef	
Parent Container	ComIPdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	References to all signal groups contained in this I-Pdu	
Multiplicity	0..*	
Type	Reference to ComSignalGroup	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_00518] Definition of EcucReferenceDef ComIPduSignalRef

Parameter Name	ComIPduSignalRef	
Parent Container	ComIPdu	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	References to all signals contained in this I-PDU.	
Multiplicity	0..*	
Type	Reference to ComSignal	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

[ECUC_Com_10021] Definition of EcucReferenceDef ComMainFunctionRouteSignalsRef

Parameter Name	ComMainFunctionRouteSignalsRef
Parent Container	ComIPdu
Upstream mapping defined	–
Description	Reference to ComMainFunctionRouteSignals which performs signal gateway related activities.
Multiplicity	0..1
Type	Reference to ComMainFunctionRouteSignals
Pre-Compile only	true

[ECUC_Com_00711] Definition of EcucReferenceDef ComPduIdRef

Parameter Name	ComPduIdRef
Parent Container	ComIPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

C.8.5 ComSignal

[ECUC_Com_00344] Definition of EcucParamConfContainerDef ComSignal

Container Name	ComSignal	
Parent Container	ComConfig	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Contains the configuration parameters of the AUTOSAR COM module's signals.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComBitPosition	1	[ECUC_Com_00259]
ComBitSize	0..1	[ECUC_Com_00158]
ComDataInvalidAction	0..1	[ECUC_Com_00314]
ComFirstTimeout	0..1	[ECUC_Com_00183]
ComHandleId	0..1	[ECUC_Com_00165]
ComInitialValueOnly	0..1	[ECUC_Com_00811]
ComRxDataTimeoutAction	0..1	[ECUC_Com_00412]
ComSignalDataInvalidValue	0..1	[ECUC_Com_00391]
ComSignalEndianness	1	[ECUC_Com_00157]
ComSignalInitValue	0..1	[ECUC_Com_00170]
ComSignalLength	0..1	[ECUC_Com_00437]
ComSignalType	1	[ECUC_Com_00127]
ComTimeout	0..1	[ECUC_Com_00263]
ComTimeoutSubstitutionValue	0..1	[ECUC_Com_10006]
ComTransferProperty	0..1	[ECUC_Com_00232]
ComUpdateBitPosition	0..1	[ECUC_Com_00257]
ComSystemTemplateSystemSignalRef	0..1	[ECUC_Com_00002]

Included Containers		
Container Name	**Multiplicity**	**Description**
ComDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameters DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
ComFilter	0..1	This container contains the configuration parameters of the AUTOSAR COM module's Filters. Note: On sender side the container is used to specify the transmission mode conditions.

For parameter table [ECUC_Com_00259] ComBitPosition, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00158] ComBitSize, see definition below container ComGroupSignal.

[ECUC_Com_00314] Definition of EcucEnumerationParamDef ComDataInvalidAction

Parameter Name	ComDataInvalidAction
Parent Container	ComSignal, ComSignalGroup
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate]
Description	This parameter defines the action performed upon reception of an invalid signal. Relating to signal groups the action in case if one of the included signals is an invalid signal. If Replace is used the ComSignalInitValue will be used for the replacement.
Multiplicity	0..1
Type	EcucEnumerationParamDef

▽

△

Range	NOTIFY	Notify (via callback)
	REPLACE	Signals shall be replaced by their ComSignalInit Value.
Pre-Compile only	true	

[ECUC_Com_00183] Definition of EcucFloatParamDef ComFirstTimeout

Parameter Name	ComFirstTimeout	
Parent Container	ComSignal, ComSignalGroup	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Defines the length of the first deadline monitoring timeout period in seconds. This timeout is used immediately after start (or restart) of the deadline monitoring service. The timeout period of the successive periods is configured by ECUC_Com_00263.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 3600]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

For parameter table [ECUC_Com_00165] ComHandleId, see definition below container ComGroupSignal.

[ECUC_Com_00811] Definition of EcucBooleanParamDef ComInitialValueOnly

Parameter Name	ComInitialValueOnly
Parent Container	ComSignal, ComSignalGroup
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This parameter defines that the respective signal's initial value shall be put into the respective PDU but there will not be any update of the value through the users (e.g. RTE, SwCluC). Thus the Com implementation does not need to expect any API calls for this signal (group).
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

[ECUC_Com_00412] Definition of EcucEnumerationParamDef ComRxDataTimeoutAction

Parameter Name	ComRxDataTimeoutAction
Parent Container	ComSignal, ComSignalGroup
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	This parameter defines the action performed upon expiration of the reception deadline monitoring timer.
Multiplicity	0..1
Type	EcucEnumerationParamDef

▽

△

Range	NONE	no replacement shall take place
	REPLACE	Signals shall be replaced by their ComSignalInit Value.
	SUBSTITUTE	signals shall be replaced by their ComTimeout SubstitutionValue
Pre-Compile only	true	

For parameter table [ECUC_Com_00391] ComSignalDataInvalidValue, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00157] ComSignalEndianness, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00170] ComSignalInitValue, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00437] ComSignalLength, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00127] ComSignalType, see definition below container ComGroupSignal.

[ECUC_Com_00263] Definition of EcucFloatParamDef ComTimeout

Parameter Name	ComTimeout	
Parent Container	ComSignal, ComSignalGroup	
Upstream mapping defined	[17, CP_TPS_SoftwareComponentTemplate], [13, CP_TPS_SystemTemplate]	
Description	Defines the length of the deadline monitoring timeout period in seconds. The period for the first timeout period can be configured separately by ECUC_Com_00183.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 3600]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

For parameter table [ECUC_Com_10006] ComTimeoutSubstitutionValue, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00232] ComTransferProperty, see definition below container ComGwDestinationDescription.

For parameter table [ECUC_Com_00257] ComUpdateBitPosition, see definition below container ComGwDestinationDescription.

For parameter table [ECUC_Com_00002] ComSystemTemplateSystemSignalRef, see definition below container ComGroupSignal.

C.8.6 ComUserModule

[ECUC_Com_10031] Definition of EcucParamConfContainerDef ComUserModule

Container Name	ComUserModule
Parent Container	ComConfig
Upstream mapping defined	–
Description	Contains the configuration parameters of the Com user modules.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComUserModuleCnfRef	1	[ECUC_Com_10029]

No Included Containers

[ECUC_Com_10029] Definition of EcucUriReferenceDef ComUserModuleCnfRef

Parameter Name	ComUserModuleCnfRef
Parent Container	ComUserModule
Upstream mapping defined	–
Description	Reference to the Com user module configuration.
Multiplicity	1
Type	Reference to destinationUri ComUserUriDefSet/ComUser
Pre-Compile only	true

C.8.7 ComUserUriDefSet

[ECUC_Com_10039] Definition of EcucDestinationUriDefSet ComUserUriDefSet

EcucDestinationUriDefSet Name	ComUserUriDefSet
Description	Defines the set of DestinationUriDefs for the Com module.
Included EcucDestinationUriDefs	
Name	**Description**
ComUser	Defines the configuration container content of the Com user modules relevant settings.

[ECUC_Com_10040] Definition of EcucDestinationUriDef ComUser

EcucDestinationUriDef Name	ComUser
Destination Uri Definition Set	ComUserUriDefSet
Description	Defines the configuration container content of the Com user modules relevant settings.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Description**
ComUserModuleCnf	0..1	Contains the configuration parameters of the Com user module.

C.8.8 ComSignalGroup

[ECUC_Com_00345] Definition of EcucParamConfContainerDef ComSignal Group

Container Name	ComSignalGroup	
Parent Container	ComConfig	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Contains the configuration parameters of the AUTOSAR COM module's signal groups.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComDataInvalidAction	0..1	[ECUC_Com_00314]
ComFirstTimeout	0..1	[ECUC_Com_00183]
ComHandleId	0..1	[ECUC_Com_00165]
ComInitialValueOnly	0..1	[ECUC_Com_00811]
ComRxDataTimeoutAction	0..1	[ECUC_Com_00412]
ComSignalGroupArrayAccess	0..1	[ECUC_Com_10003]
ComTimeout	0..1	[ECUC_Com_00263]
ComTransferProperty	0..1	[ECUC_Com_00232]
ComUpdateBitPosition	0..1	[ECUC_Com_00257]
ComSystemTemplateSignalGroupRef	0..1	[ECUC_Com_00001]

Included Containers		
Container Name	**Multiplicity**	**Description**
ComDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameters DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
ComGroupSignal	0..*	This container contains the configuration parameters of group signals. I.e. signals that are included within a signal group.

For parameter table [ECUC_Com_00314] ComDataInvalidAction, see definition below container ComSignal.

For parameter table [ECUC_Com_00183] ComFirstTimeout, see definition below container ComSignal.

For parameter table [ECUC_Com_00165] ComHandleId, see definition below container ComGroupSignal.

For parameter table [ECUC_Com_00811] ComInitialValueOnly, see definition below container ComSignal.

For parameter table [ECUC_Com_00412] ComRxDataTimeoutAction, see definition below container ComSignal.

[ECUC_Com_10003] Definition of EcucBooleanParamDef ComSignalGroupArrayAccess

Parameter Name	ComSignalGroupArrayAccess
Parent Container	ComSignalGroup
Upstream mapping defined	[13, CP_TPS_SystemTemplate]
Description	Defines whether the uint8-array based access shall be used for this ComSignalGroup.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

For parameter table [ECUC_Com_00263] ComTimeout, see definition below container ComSignal.

For parameter table [ECUC_Com_00232] ComTransferProperty, see definition below container ComGwDestinationDescription.

For parameter table [ECUC_Com_00257] ComUpdateBitPosition, see definition below container ComGwDestinationDescription.

[ECUC_Com_00001] Definition of EcucForeignReferenceDef ComSystemTemplateSignalGroupRef

Parameter Name	ComSystemTemplateSignalGroupRef	
Parent Container	ComSignalGroup	
Upstream mapping defined	[13, CP_TPS_SystemTemplate]	
Description	Reference to the ISignalToIPduMapping that contains a reference to the ISignalGroup (SystemTemplate) which this ComSignalGroup represents.	
Multiplicity	0..1	
Type	Foreign reference to I-SIGNAL-TO-I-PDU-MAPPING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

C.9 WdgM

C.9.1 WdgMTransitionProxy

[ECUC_WdgM_00364] Definition of EcucParamConfContainerDef WdgMTransitionProxy

Container Name	WdgMTransitionProxy
Parent Container	WdgMExternalLogicalSupervision
Upstream mapping defined	–
Description	The WdgMTransitionProxy defines a proxy for a transition between the Host Software Cluster and an Application Software Cluster and vice versa. From the Host Software Cluster perspective a Cross Cluster Transition graph leaves the host after the transition which has the WdgMTransitionProxy as a destination or initial reference and returns in this WdgMTransitionProxy after the configured transitions are occurred in the related Application Software Cluster. Afterwards the transition in the host are expected which are referencing the WdgMTransitionProxy by a source or final reference.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
WdgMResourceRef	1	[ECUC_WdgM_00367]

No Included Containers

For parameter table [ECUC_WdgM_00367] WdgMResourceRef, see definition below container WdgMBaseSocket.

C.9.2 WdgMBaseSocket

[ECUC_WdgM_00377] Definition of EcucParamConfContainerDef WdgMBaseSocket

Container Name	WdgMBaseSocket
Parent Container	WdgMGeneral
Upstream mapping defined	–
Description	This container configures how many EcucPartitions specific infrastructure links are required for the WdgM instances in Application Software Clusters provided by the Host Software Cluster. Such infrastructure links serve for: the initialization of Application Software Cluster WdgM instances by Host WdgM instance the transmission of supervision results from Application Software Cluster WdgM instances to Host WdgM instance any other implementation specific purpose which is need for the interaction of Application Software Cluster WdgM instances and Host WdgM instance If the infrastructure connection is specific to one or several EcucPartition(s) the Wdg MSocketEcucPartitionRef(s) denotes the applicable EcucPartition.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
WdgMResourceRef	1	[ECUC_WdgM_00367]
WdgMSocketEcucPartitionRef	0..*	[ECUC_WdgM_00366]

No Included Containers

[ECUC_WdgM_00367] Definition of EcucForeignReferenceDef WdgMResourceRef

Parameter Name	WdgMResourceRef
Parent Container	WdgMBaseSocket, WdgMTransitionProxy
Upstream mapping defined	–
Description	Reference to the CpSoftwareClusterServiceResource.
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Pre-Compile only	true

[ECUC_WdgM_00366] Definition of EcucReferenceDef WdgMSocketEcucPartitionRef

Parameter Name	WdgMSocketEcucPartitionRef
Parent Container	WdgMBaseSocket
Upstream mapping defined	–
Description	Reference to the EcucPartition.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

D Referenced C-API

D.1 RTE

D.1.1 RTE Lifecycle API Reference

D.1.1.1 Rte_Init

[SWS_Rte_91142] Definition of API function Rte_Init_<InitContainer>

Service Name	Rte_Init_<InitContainer>
Syntax	`void Rte_Init_<InitContainer> ( void )`
Service ID [hex]	0x75
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Init is intended schedule RunnableEntitys for initialization purpose which are mapped to the related RteInitializationRunnableBatch container.
Available via	Rte.h

D.1.1.2 Rte_Start

[SWS_Rte_91137] Definition of API function Rte_Start_<EcucPartition>

Service Name	Rte_Start_<EcucPartition>	
Syntax	`Std_ReturnType Rte_Start_<EcucPartition> ( void )`	
Service ID [hex]	0x70	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`RTE_E_OK`: No error occurred. `RTE_E_LIMIT`: An internal limit has been exceeded. The allocation of a required resource has failed.
Description	Rte_Start is intended to allocate and initialize system resources and communication resources used by the RTE in the given partition.	
Available via	Rte.h	

D.1.2 RTE RIPS API Reference

D.1.2.1 Rte_Rips_DataIsUpdated

[SWS_Rte_91119] Definition of API function Rte_Rips_<PlugIn>_DataIsUpdated_<SwcBswI>_<CGI>

Service Name	Rte_Rips_<PlugIn>_DataIsUpdated_<SwcBswI>_<CGI>	
Syntax	`boolean Rte_Rips_<PlugIn>_DataIsUpdated_<SwcBswI>_<CGI> ( void )`	
Service ID [hex]	0xB4	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	The return value is used to indicate if the data has been updated or not.
Description	The Rte_Rips_DataIsUpdated API provides access to the update flag for an explicit receiver	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.2 Rte_Rips_DRead

[SWS_Rte_91122] Definition of API function Rte_Rips_<PlugIn>_DRead_<SwcBswI>[Partition][_<ExE>]_<CGI>

Service Name	Rte_Rips_<PlugIn>_DRead_<SwcBswI>[Partition][_<ExE>]_<CGI>	
Syntax	`<return> Rte_Rips_<PlugIn>_DRead_<SwcBswI>[Partition][_<ExE>]_<CGI> ( void )`	
Service ID [hex]	0xFF	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	<return>	Return value provides access to the data value of the Variable DataPrototype. For details of the <return> value definition see section 5.2.6.6.
Description	Rte_Rips_DRead Performs an "explicit" read on a sender-receiver communication data element typed by a primitive data type.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.3 Rte_Rips_DataIsUpdated_EventActivation

[SWS_Rte_91000] Definition of API function Rte_Rips_<PlugIn>_DataIsUpdated EventActivation_<SwcBswI>_<DR>_<CGI>

Service Name	Rte_Rips_<PlugIn>_DataIsUpdatedEventActivation_<SwcBswI>_<DR>_<CGI>	
Syntax	`boolean Rte_Rips_<PlugIn>_DataIsUpdatedEventActivation_<SwcBsw I>_<DR>_<CGI> ( void )`	
Service ID [hex]	0xB5	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	The return value is used to indicate if the Runnable shall be activated or not.
Description	The Rte_Rips_DataIsUpdated_EventActivation API indicates the RTE to activate the Runnable Entity triggered by DataReceivedEvent when the related VariableDataPrototype has been updated	
Available via	Rte_Rips_<PlugIn>.h	

D.1.2.4 Rte_Rips_Feedback

[SWS_Rte_91121] Definition of API function Rte_Rips_<PlugIn>_Feedback_[<SwcBswI>][Partition]_<CGI>

Service Name	Rte_Rips_<PlugIn>_Feedback_[<SwcBswI>][Partition]_<CGI>	
Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Feedback_[<SwcBswI>][Partition]_<CGI> ( void )`	
Service ID [hex]	0xB6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	The return value is used to pass error notifications.
Description	The Rte_Rips_Feedback API provides access to acknowledgment notifications for explicit and implicit sender-receiver communication and to pass error notification to senders in a Software Cluster	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.5 Rte_Rips_Invoke

[SWS_Rte_89022] Definition of API function Rte_Rips_<PlugIn>_Invoke_<SwcBswI>_<CGI>

Service Name	Rte_Rips_<PlugIn>_Invoke_<SwcBswI>_<CGI>	
Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Invoke_<SwcBswI>_<CGI> ( [IN|IN/OUT|OUT] <data_1>, [IN|IN/OUT|OUT] ..., [IN|IN/OUT|OUT] <data_n>, [Std_TransformerError transformerError] )`	
Service ID [hex]	0xEC	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	<data_1>	The Rte_Rips_Invoke API includes zero or more IN, IN/OUT and OUT parameters according [SWS_Rte_01102] and none in case of triggers
Parameters (inout)	...	The Rte_Rips_Invoke API includes zero or more IN, IN/OUT and OUT parameters according [SWS_Rte_01102] and none in case of triggers
Parameters (out)	<data_n>	The Rte_Rips_Invoke API includes zero or more IN, IN/OUT and OUT parameters according [SWS_Rte_01102] and none in case of triggers
	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors.
Description	Rte_Rips_Invoke performs a transformer or cross cluster invocation for clients or trigger sources.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.6 Rte_Rips_InvocationHandler

[SWS_Rte_89015] Definition of API function <name of the Invocation Handler>

Service Name	<name of the Invocation Handler>
Syntax	`void <name of the Invocation Handler> ( void )`
Service ID [hex]	0xEE
Sync/Async	Synchronous
Reentrancy	Conditional Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Performs invocation of server runnables, hard error runnables, ASCR runnables and triggered runnables via a transformer".
Available via	Rte_Rips_<PlugIn>.h

D.1.2.7 Rte_Rips_Prm

[SWS_Rte_91116] Definition of API function Rte_Rips_<PlugIn>_Prm_<CGI>

Service Name	Rte_Rips_<PlugIn>_Prm_<CGI>	

Syntax	`<return> Rte_Rips_<PlugIn>_Prm_<CGI> (`
	`  void`
	`)`	

Service ID [hex]	0x0100	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	<return>	For primitive data types, the Rte_Rips_Prm API returns the parameter value. For composite data types, the Rte_Rips_Prm API returns a reference (in C, a pointer) to the constant parameter.
Description	The Rte_Rips_Prm API provides access to a parameter provided by another Software Cluster	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.8 Rte_Rips_Read

[SWS_Rte_89020] Definition of API function Rte_Rips_<PlugIn>_Read_[<SwcBswI>][Partition][_<ExE>]_<CGI>

Service Name	Rte_Rips_<PlugIn>_Read_[<SwcBswI>][Partition][_<ExE>]_<CGI>	

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Read_[<SwcBswI>][Partition][_<ExE>]_<CGI> (`
	`  OUT <data>,`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xEA	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	<data>	The OUT parameter <data> pass back the received data.
	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors.
Description	Rte_Rips_Read Performs an "explicit" read on a sender-receiver communication data element.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.9 Rte_Rips_ReturnResult

[SWS_Rte_89023] Definition of API function Rte_Rips_<PlugIn>_ReturnResult_<SwcBswI>_<CGI>

Service Name	Rte_Rips_<PlugIn>_ReturnResult_<SwcBswI>_<CGI>	

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_ReturnResult_<SwcBswI>_<CGI> (`
	`  [IN/OUT|OUT] <param_1>,`
	`  [IN/OUT|OUT] <param_n>,`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xED	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	<param_1>	The Rte_Rips_ReturnResult API includes zero or more IN/OUT and OUT parameters according [SWS_Rte_01111].
Parameters (out)	<param_n>	The Rte_Rips_ReturnResult API includes zero or more IN/OUT and OUT parameters according [SWS_Rte_01111].
	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors
Description	Rte_Rips_ReturnResult get the server results of a performed a transformer or cross cluster invocation for clients.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.2.10 Rte_Rips_Start

[SWS_Rte_89017] Definition of API function Rte_Rips_<PlugIn>_Rte_Start

Service Name	Rte_Rips_<PlugIn>_Rte_Start

Syntax	`void Rte_Rips_<PlugIn>_Rte_Start (`
	`  void`
	`)`

Service ID [hex]	0xF1
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_Rte_Start initializes those RTE Implementation Plug-In parts which are relevant for the RTE related operation.
Available via	Rte_Rips_<PlugIn>.h

D.1.2.11 Rte_Rips_Stop

[SWS_Rte_89018] Definition of API function Rte_Rips_<PlugIn>_Rte_Stop

Service Name	Rte_Rips_<PlugIn>_Rte_Stop

Syntax	`void Rte_Rips_<PlugIn>_Rte_Stop (`
	`  void`
	`)`

Service ID [hex]	0xF2
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_Rte_Stop deinitializes those RTE Implementation Plug-In parts which are relevant for the RTE related operation.
Available via	Rte_Rips_<PlugIn>.h

D.1.2.12 Rte_Rips_SchM_Deinit

[SWS_Rte_89019] Definition of API function Rte_Rips_SchM_Deinit

Service Name	Rte_Rips_SchM_Deinit

Syntax	`void Rte_Rips_SchM_Deinit (`
	`  void`
	`)`

Service ID [hex]	0xF3
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_SchM_Deinit deinitializes those RTE Implementation Plug-In parts which are relevant for the SchM related operations.
Available via	Rte_Rips_<PlugIn>.h

D.1.2.13 Rte_Rips_SchM_Init

[SWS_Rte_89016] Definition of API function Rte_Rips_<PlugIn>_SchM_Init

Service Name	Rte_Rips_<PlugIn>_SchM_Init

Syntax	`void Rte_Rips_<PlugIn>_SchM_Init (`
	`  void`
	`)`

Service ID [hex]	0xF0
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_SchM_Init initializes those RTE Implementation Plug-In parts which are relevant for the SchM related operations.
Available via	Rte_Rips_<PlugIn>.h

D.1.2.14 Rte_Rips_SwitchNotificationStatusType

[SWS_Rte_91113] Definition of datatype Rte_Rips_SwitchNotificationStatusType

Name	Rte_Rips_SwitchNotificationStatusType		
Kind	Type		
Derived from	uint8		
Range	RTE_SWITCH_ NOTIFICATION_SKIP	0x00	mode switch notification cannot be dequeued
	RTE_SWITCH_ NOTIFICATION_ ENQUEUED_FIRST	0x01	mode switch notification is enqueued into an empty mode queue
	RTE_SWITCH_ NOTIFICATION_ ENQUEUED_NOT_FIRST	0x02	mode switch notification is enqueued into a non empty mode queue
	RTE_SWITCH_ NOTIFICATION_ ENQUEUE_FAILED	0x03	enqueue operation into a non empty mode queue failed
	RTE_SWITCH_ NOTIFICATION_ DEQUEUED_LAST	0x04	last mode switch notification was enqueued from mode queue
	RTE_SWITCH_ NOTIFICATION_ DEQUEUED_NOT_LAST	0x05	mode switch notification was enqueued from mode queue, further mode switch notifications are in the queue
Description	Status of the en- and dequeue operation on a mode queue		
Available via	Rte_Type.h		

D.1.2.15 Rte_Rips_Switch

[SWS_Rte_91114] Definition of API function Rte_Rips_<PlugIn>_Switch_<BswSwcI>_<MMI>

Service Name	Rte_Rips_<PlugIn>_Switch_<BswSwcI>_<MMI>	
Syntax	`void Rte_Rips_<PlugIn>_Switch_<BswSwcI>_<MMI> ( Rte_Rips_SwitchNotificationStatusType switchNotificationStatus, uint32 previousmode, uint32 nextmode )`	
Service ID [hex]	0xB0	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	switchNotificationStatus	Status of the enqueue operation
	previousmode	The value of the ModeDeclaration of the mode being left
	nextmode	The value of the ModeDeclaration of the mode being entered
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Rte_Rips_StartModeSwitch notifies the RTE Implementation Plug-In about an enqueue operation in a mode queue.	
Available via	Rte_Rips_<PlugIn>.h	

D.1.2.16 Rte_Rips_DequeueModeSwitch

[SWS_Rte_91115] Definition of API function Rte_Rips_<PlugIn>_DequeueModeSwitch_<MMI>_<OsTask>

Service Name	Rte_Rips_<PlugIn>_DequeueModeSwitch_<MMI>_<OsTask>	
Syntax	`Rte_Rips_SwitchNotificationStatusType Rte_Rips_<PlugIn>_DequeueModeSwitch_<MMI>_<OsTask> ( void )`	
Service ID [hex]	0xB1	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Rte_Rips_SwitchNotificationStatusType	The return value is used indicate the status of the dequeue operation in a mode queue
Description	Rte_Rips_DequeueModeSwitch dequeues a mode switch notification from the mode queue when it is called after the last on-entry ExecutableEntity terminated.	
Available via	Rte_Buffers.h	

D.1.2.17 Rte_Rips_Trigger

[SWS_Rte_91117] Definition of API function Rte_Rips_<PlugIn>_Trigger_<BswSwcI>_<MMI>

Service Name	Rte_Rips_<PlugIn>_Trigger_<BswSwcI>_<MMI>

Syntax	`void Rte_Rips_<PlugIn>_Trigger_<BswSwcI>_<MMI> (`
	`  void`
	`)`

Service ID [hex]	0xB2
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_Trigger notifies the RTE Implementation Plug-In about a raised trigger.
Available via	Rte_Rips_<PlugIn>.h

D.1.2.18 Rte_Rips_Write

[SWS_Rte_89021] Definition of API function Rte_Rips_<PlugIn>_Write_[<SwcBswI>][Partition][_<ExE>]_<CGI>

Service Name	Rte_Rips_<PlugIn>_Write_[<SwcBswI>][Partition][_<ExE>]_<CGI>	

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Write_[<SwcBswI>][Partition][_<ExE>]_<CGI> (`
	`  IN <data>,`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xEB	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	<data>	The IN parameter <data> pass the received data.
Parameters (inout)	None	
Parameters (out)	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors.
Description	Rte_Rips_Write Performs an "explicit" write on a sender-receiver communication data element.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

D.1.3 RTE API Reference

See document [4]:

- `Rte_Read`
- `Rte_DRead`

- Rte_Write
- Rte_Call
- Rte_Switch
- Rte_Trigger

D.2 OS

See document [5] and [24] as reference for Os.

- TASK
- ActivateTask
- ChainTask
- TerminateTask
- GetResource
- ReleaseResource
- SuspendOSInterrupts
- ResumeOSInterrupts
- GetSpinlock
- ReleaseSpinlock
- GetApplicationID
- StartScheduleTable
- StopScheduleTable
- Schedule
- DisableAllInterrupts
- EnableAllInterrupts
- SuspendAllInterrupts
- ResumeAllInterrupts
- CallTrustedFunction
- GetCoreID

D.3 NvM

D.3.1 NvM_CancelJobs

[SWS_NvM_00535] Definition of API function NvM_CancelJobs

Service Name	NvM_CancelJobs	
Syntax	`Std_ReturnType NvM_CancelJobs ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The job was successfully removed from queue. `E_NOT_OK`: The job could not be found in the queue.
Description	Service to cancel all jobs pending for a NV block.	
Available via	NvM.h	

D.3.2 NvM_EraseNvBlock

[SWS_NvM_00457] Definition of API function NvM_EraseNvBlock

Service Name	NvM_EraseNvBlock	
Syntax	`Std_ReturnType NvM_EraseNvBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x09	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to erase a NV block.	
Available via	NvM.h	

D.3.3 NvM_GetDataIndex

[SWS_NvM_00449] Definition of API function NvM_GetDataIndex

Service Name	NvM_GetDataIndex	
Syntax	`Std_ReturnType NvM_GetDataIndex ( NvM_BlockIdType BlockId, uint8* DataIndexPtr )`	
Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	DataIndexPtr	Pointer to where to store the current dataset index (0..255)
Return value	Std_ReturnType	`E_OK`: The index position has been retrieved successfully. `E_NOT_OK`: An error occurred.
Description	Service for getting the currently set DataIndex of a dataset NVRAM block	
Available via	NvM.h	

D.3.4 NvM_GetErrorStatus

[SWS_NvM_00451] Definition of API function NvM_GetErrorStatus

Service Name	NvM_GetErrorStatus	
Syntax	`Std_ReturnType NvM_GetErrorStatus ( NvM_BlockIdType BlockId, NvM_RequestResultType* RequestResultPtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	RequestResultPtr	Pointer to where to store the request result. See NvM_Request ResultType .
Return value	Std_ReturnType	`E_OK`: The block dependent error/status information was read successfully. `E_NOT_OK`: An error occured.
Description	Service to read the block dependent error/status information.	
Available via	NvM.h	

D.3.5 NvM_InvalidateNvBlock

[SWS_NvM_00459] Definition of API function NvM_InvalidateNvBlock

Service Name	NvM_InvalidateNvBlock	
Syntax	`Std_ReturnType NvM_InvalidateNvBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x0b	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to invalidate a NV block.	
Available via	NvM.h	

D.3.6 NvM_ReadBlock

[SWS_NvM_00454] Definition of API function NvM_ReadBlock

Service Name	NvM_ReadBlock	
Syntax	`Std_ReturnType NvM_ReadBlock ( NvM_BlockIdType BlockId, void* NvM_DstPtr )`	
Service ID [hex]	0x06	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	NvM_DstPtr	Pointer to the RAM data block.
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to copy the data of the NV block to its corresponding RAM block.	
Available via	NvM.h	

D.3.7 NvM_ReadPRAMBlock

[SWS_NvM_00764] Definition of API function NvM_ReadPRAMBlock

Service Name	NvM_ReadPRAMBlock	
Syntax	`Std_ReturnType NvM_ReadPRAMBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x16	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to copy the data of the NV block to its corresponding permanent RAM block.	
Available via	NvM.h	

D.3.8 NvM_RestoreBlockDefaults

[SWS_NvM_00456] Definition of API function NvM_RestoreBlockDefaults

Service Name	NvM_RestoreBlockDefaults	
Syntax	`Std_ReturnType NvM_RestoreBlockDefaults ( NvM_BlockIdType BlockId, void* NvM_DestPtr )`	
Service ID [hex]	0x08	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	NvM_DestPtr	Pointer to the RAM data block.
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to restore the default data to its corresponding RAM block.	
Available via	NvM.h	

D.3.9 NvM_RestorePRAMBlockDefaults

[SWS_NvM_00813] Definition of API function NvM_RestorePRAMBlockDefaults

Service Name	NvM_RestorePRAMBlockDefaults	
Syntax	`Std_ReturnType NvM_RestorePRAMBlockDefaults ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x18	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to restore the default data to its corresponding permanent RAM block.	
Available via	NvM.h	

D.3.10 NvM_SetBlockProtection

[SWS_NvM_00450] Definition of API function NvM_SetBlockProtection

Service Name	NvM_SetBlockProtection	
Syntax	`Std_ReturnType NvM_SetBlockProtection ( NvM_BlockIdType BlockId, boolean ProtectionEnabled )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	ProtectionEnabled	TRUE: Write protection shall be enabled FALSE: Write protection shall be disabled
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The block was enabled/disabled as requested `E_NOT_OK`: An error occured.
Description	Service for setting/resetting the write protection for a NV block.	
Available via	NvM.h	

D.3.11 NvM_SetDataIndex

[SWS_NvM_00448] Definition of API function NvM_SetDataIndex

Service Name	NvM_SetDataIndex	
Syntax	`Std_ReturnType NvM_SetDataIndex ( NvM_BlockIdType BlockId, uint8 DataIndex )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	DataIndex	Index position (association) of a NV/ROM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The index position was set successfully. `E_NOT_OK`: An error occurred.
Description	Service for setting the DataIndex of a dataset NVRAM block.	
Available via	NvM.h	

D.3.12 NvM_SetRamBlockStatus

[SWS_NvM_00453] Definition of API function NvM_SetRamBlockStatus

Service Name	NvM_SetRamBlockStatus	
Syntax	`Std_ReturnType NvM_SetRamBlockStatus ( NvM_BlockIdType BlockId, boolean BlockChanged )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	BlockChanged	TRUE: Validate the permanent RAM block or the explicit synchronization and mark block as changed. FALSE: Invalidate the permanent RAM block or the explicit synchronization and mark block as unchanged.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The status of the permanent RAM block or the explicit synchronization was changed as requested. `E_NOT_OK`: An error occurred.
Description	Service for setting the RAM block status of a permanent RAM block or the status of the explicit synchronization of a NVRAM block.	
Available via	NvM.h	

D.3.13 NvM_WriteBlock

[SWS_NvM_00455] Definition of API function NvM_WriteBlock

Service Name	NvM_WriteBlock	
Syntax	`Std_ReturnType NvM_WriteBlock ( NvM_BlockIdType BlockId, const void* NvM_SrcPtr )`	
Service ID [hex]	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	NvM_SrcPtr	Pointer to the RAM data block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to copy the data of the RAM block to its corresponding NV block.	
Available via	NvM.h	

D.3.14 NvM_WritePRAMBlock

[SWS_NvM_00793] Definition of API function NvM_WritePRAMBlock

Service Name	NvM_WritePRAMBlock	
Syntax	`Std_ReturnType NvM_WritePRAMBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x17	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: request has been accepted `E_NOT_OK`: request has not been accepted
Description	Service to copy the data of the permanent RAM block to its corresponding NV block.	
Available via	NvM.h	

D.3.15 NvM_SingleBlockCallbackFunction

[SWS_NvM_00467] Definition of configurable interface NvM_SingleBlockCallbackFunction

Service Name	NvM_SingleBlockCallbackFunction	
Syntax	`Std_ReturnType NvM_SingleBlockCallbackFunction ( NvM_BlockRequestType BlockRequest, NvM_RequestResultType JobResult )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockRequest	The request type (read, write, ... etc.) of the previous processed block job
	JobResult	The request result of the previous processed block job.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Per block callback routine to notify the upper layer that an asynchronous single block request has been finished.	
Available via	If NvMBlockHeaderInclude is configured NvM will include this to get the prototype, otherwise NvM provides NvM_Externals.h	

D.3.16 NvM_InitBlockCallbackFunction

[SWS_NvM_00469] Definition of configurable interface NvM_InitBlockCallbackFunction

Service Name	NvM_InitBlockCallbackFunction	
Syntax	`Std_ReturnType NvM_InitBlockCallbackFunction ( NvM_InitBlockRequestType InitBlockRequest )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	InitBlockRequest	The request type (read, restore, ... etc.) of the currently processed block
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Per block callback routine which shall be called by the NvM module when default data needs to be restored in RAM, and a ROM block is not configured.	
Available via	If NvMBlockHeaderInclude is configured NvM will include this to get the prototype, otherwise NvM provides NvM_Externals.h	

D.3.17 NvM_ReadRamBlockFromNvm

[SWS_NvM_00540] Definition of configurable interface NvM_ReadRamBlockFromNvm

Service Name	NvM_ReadRamBlockFromNvm	
Syntax	`Std_ReturnType NvM_ReadRamBlockFromNvm ( const void* NvMBuffer )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	NvMBuffer	the address of the buffer where the data can be read from
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Block specific callback routine which shall be called in order to let the application copy data from NvM module's mirror to RAM block.	
Available via	If NvMBlockHeaderInclude is configured NvM will include this to get the prototype, otherwise Nv M provides NvM_Externals.h	

D.3.18 NvM_WriteRamBlockToNvm

[SWS_NvM_00539] Definition of configurable interface NvM_WriteRamBlockTo Nvm

Service Name	NvM_WriteRamBlockToNvm	
Syntax	`Std_ReturnType NvM_WriteRamBlockToNvm ( void* NvMBuffer )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	NvMBuffer	the address of the buffer where the data shall be written to
Return value	Std_ReturnType	`E_OK`: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Block specific callback routine which shall be called in order to let the application copy data from RAM block to NvM module's mirror.	
Available via	If NvMBlockHeaderInclude is configured NvM will include this to get the prototype, otherwise Nv M provides NvM_Externals.h	

D.3.19 NvM_ReadAll

[SWS_NvM_00460] Definition of API function NvM_ReadAll

Service Name	NvM_ReadAll
Syntax	`void NvM_ReadAll ( void )`
Service ID [hex]	0x0c
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Initiates a multi block read request.
Available via	NvM.h

D.3.20 NvM_WriteAll

[SWS_NvM_00461] Definition of API function NvM_WriteAll

Service Name	NvM_WriteAll
Syntax	`void NvM_WriteAll ( void )`
Service ID [hex]	0x0d
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Initiates a multi block write request.
Available via	NvM.h

D.4 Com

D.4.1 APIs to handle ComSignals / ComSignalGroups

D.4.1.1 Com_SendSignal

[SWS_Com_00197] Definition of API function Com_SendSignal

Service Name	Com_SendSignal	
Syntax	`uint8 Com_SendSignal ( Com_SignalIdType SignalId, const void* SignalDataPtr )`	
Service ID [hex]	0x0a	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be sent.
	SignalDataPtr	Reference to the signal data to be transmitted.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	`E_OK`: service has been accepted `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group was stopped `COM_BUSY`: in case the TP-Buffer is locked for large data types handling
Description	The service Com_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.	
Available via	Com.h	

D.4.1.2 Com_SendDynSignal

[SWS_Com_00627] Definition of API function Com_SendDynSignal

Service Name	Com_SendDynSignal	
Syntax	`uint8 Com_SendDynSignal ( Com_SignalIdType SignalId, const void* SignalDataPtr, uint32 Length )`	
Service ID [hex]	0x21	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be sent.
	SignalDataPtr	Reference to the signal data to be transmitted.
	Length	Length of the dynamic length signal
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	uint8	E_OK: service has been accepted E_NOT_OK: in case the Length is greater than the configured ComSignalLength of this sent signal COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked
Description	The service Com_SendDynSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.	
Available via	Com.h	

D.4.1.3 Com_SendSignalGroup

[SWS_Com_00200] Definition of API function Com_SendSignalGroup

Service Name	Com_SendSignalGroup	

Syntax	`uint8 Com_SendSignalGroup (`
	`  Com_SignalGroupIdType SignalGroupId`
	`)`	

Service ID [hex]	0x0d	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be sent.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_SendSignalGroup copies the content of the associated shadow buffer to the associated I-PDU.	
Available via	Com.h	

D.4.1.4 Com_SendSignalGroupArray

[SWS_Com_00851] Definition of API function Com_SendSignalGroupArray

Service Name	Com_SendSignalGroupArray	

Syntax	`uint8 Com_SendSignalGroupArray (`
	`  Com_SignalGroupIdType SignalGroupId,`
	`  const uint8* SignalGroupArrayPtr`
	`)`	

Service ID [hex]	0x23	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be sent.

▽

△

	SignalGroupArrayPtr	Reference to the signal group array.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_SendSignalGroupArray copies the content of the provided SignalGroupArray Ptr to the associated I-PDU. The provided data shall correspond to the array representation of the signal group.	
Available via	Com.h	

D.4.1.5 Com_SendSignalWithMetaData

[SWS_Com_91009] Definition of API function Com_SendSignalWithMetaData

Service Name	Com_SendSignalWithMetaData	
Syntax	`uint8 Com_SendSignalWithMetaData ( Com_SignalIdType SignalId, const void* SignalDataPtr, const uint8* MetaDataPtr )`	
Service ID [hex]	0x4d	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be sent.
	SignalDataPtr	Reference to the signal data to be transmitted.
	MetaDataPtr	Pointer to the meta data of the signal
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_SendSignalWithMetaData updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter. The meta data is provided via the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.6 Com_SendDynSignalWithMetaData

[SWS_Com_91010] Definition of API function Com_SendDynSignalWithMetaData

Service Name	Com_SendDynSignalWithMetaData	
Syntax	`uint8 Com_SendDynSignalWithMetaData ( Com_SignalIdType SignalId, const void* SignalDataPtr, uint32 Length, const uint8* MetaDataPtr )`	
Service ID [hex]	0x4e	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be sent.
	SignalDataPtr	Reference to the signal data to be transmitted.
	Length	Length of the dynamic length signal
	MetaDataPtr	Pointer to the meta data of the signal
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	`E_OK`: service has been accepted `E_NOT_OK`: in case the Length is greater than the configured ComSignalLength of this sent signal `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group was stopped `COM_BUSY`: in case the TP-Buffer is locked
Description	The service Com_SendDynSignalWithMetaData updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter. The meta data is provided via the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.7 Com_SendSignalGroupWithMetaData

[SWS_Com_91011] Definition of API function Com_SendSignalGroupWithMetaData

Service Name	Com_SendSignalGroupWithMetaData	
Syntax	`uint8 Com_SendSignalGroupWithMetaData ( Com_SignalGroupIdType SignalGroupId, const uint8* MetaDataPtr )`	
Service ID [hex]	0x4f	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be sent.
	MetaDataPtr	Pointer to the meta data of the signal group
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_SendSignalGroupWithMetaData copies the content of the associated shadow buffer to the associated I-PDU. The meta data is provided via the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.8 Com_SendSignalGroupArrayWithMetaData

[SWS_Com_91012] Definition of API function Com_SendSignalGroupArrayWithMetaData

Service Name	Com_SendSignalGroupArrayWithMetaData	

Syntax	`uint8 Com_SendSignalGroupArrayWithMetaData (`
	`  Com_SignalGroupIdType SignalGroupId,`
	`  const uint8* SignalGroupArrayPtr,`
	`  const uint8* MetaDataPtr`
	`)`	

Service ID [hex]	0x50	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be sent.
	SignalGroupArrayPtr	Reference to the signal group array.
	MetaDataPtr	Pointer to the meta data of the signal group.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_SendSignalGroupArrayWithMetaData copies the content of the provided SignalGroupArrayPtr to the associated I-PDU. The provided data shall correspond to the array representation of the signal group. The meta data is provided via the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.9 Com_ReceiveSignal

[SWS_Com_00198] Definition of API function Com_ReceiveSignal

Service Name	Com_ReceiveSignal	
Syntax	`uint8 Com_ReceiveSignal ( Com_SignalIdType SignalId, void* SignalDataPtr )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be received.
Parameters (inout)	None	
Parameters (out)	SignalDataPtr	Reference to the location where the received signal data shall be stored
Return value	uint8	`E_OK`: service has been accepted `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group was stopped `COM_BUSY`: in case the TP-Buffer is locked for large data types handling
Description	Com_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.	
Available via	Com.h	

D.4.1.10 Com_ReceiveDynSignal

[SWS_Com_00690] Definition of API function Com_ReceiveDynSignal

Service Name	Com_ReceiveDynSignal	
Syntax	`uint8 Com_ReceiveDynSignal ( Com_SignalIdType SignalId, void* SignalDataPtr, uint32* Length )`	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be received.
Parameters (inout)	Length	in: maximum length that could be received out: length of the dynamic length signal
Parameters (out)	SignalDataPtr	reference to the location where the received signal data shall be stored
Return value	uint8	`E_OK`: service has been accepted `E_NOT_OK`: in case the Length (as in-parameter) is smaller than the received length of the dynamic length signal `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group was stopped `COM_BUSY`: in case the TP-Buffer is locked

▽

△

Description	Com_ReceiveDynSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr and stores the length of the dynamical length signal at the position given by the Length parameter.
Available via	Com.h

D.4.1.11 Com_ReceiveSignalGroup

[SWS_Com_00201] Definition of API function Com_ReceiveSignalGroup

Service Name	Com_ReceiveSignalGroup	
Syntax	`uint8 Com_ReceiveSignalGroup ( Com_SignalGroupIdType SignalGroupId )`	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be received.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	`E_OK`: service has been accepted `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group was stopped `COM_BUSY`: in case the TP-Buffer is locked for large data types handling
Description	The service Com_ReceiveSignalGroup copies the received signal group from the I-PDU to the shadow buffer.	
Available via	Com.h	

D.4.1.12 Com_ReceiveSignalGroupArray

[SWS_Com_00854] Definition of API function Com_ReceiveSignalGroupArray

Service Name	Com_ReceiveSignalGroupArray	
Syntax	`uint8 Com_ReceiveSignalGroupArray ( Com_SignalGroupIdType SignalGroupId, uint8* SignalGroupArrayPtr )`	
Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be received.
Parameters (inout)	None	
Parameters (out)	SignalGroupArrayPtr	reference to the location where the received signal group array shall be stored

▽

△

Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_ReceiveSignalGroupArray copies the received signal group array representation from the PDU to the location designated by SignalGroupAr-rayPtr.	
Available via	Com.h	

D.4.1.13 Com_ReceiveSignalWithMetaData

[SWS_Com_91005] Definition of API function Com_ReceiveSignalWithMetaData

Service Name	Com_ReceiveSignalWithMetaData	
Syntax	`uint8 Com_ReceiveSignalWithMetaData ( Com_SignalIdType SignalId, void* SignalDataPtr, uint8* MetaDataPtr )`	
Service ID [hex]	0x49	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be received.
Parameters (inout)	None	
Parameters (out)	SignalDataPtr	Reference to the location where the received signal data shall be stored
	MetaDataPtr	Pointer to the meta data of the signal
Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	Com_ReceiveSignalWithMetaData copies the data of the signal identified by SignalId to the location specified by SignalDataPtr. The received meta data is stored at the position given by the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.14 Com_ReceiveDynSignalWithMetaData

[SWS_Com_91006] Definition of API function Com_ReceiveDynSignalWithMetaData

Service Name	Com_ReceiveDynSignalWithMetaData	
Syntax	`uint8 Com_ReceiveDynSignalWithMetaData ( Com_SignalIdType SignalId, void* SignalDataPtr, uint32* Length, uint8* MetaDataPtr )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be received.
Parameters (inout)	Length	in: maximum length that could be received out: length of the dynamic length signal
Parameters (out)	SignalDataPtr	reference to the location where the received signal data shall be stored
	MetaDataPtr	Pointer to the meta data of the signal
Return value	uint8	`E_OK`: service has been accepted `E_NOT_OK`: in case the Length (as in-parameter) is smaller than the received length of the dynamic length signal `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group was stopped `COM_BUSY`: in case the TP-Buffer is locked
Description	Com_ReceiveDynSignalWithMetaData copies the data of the signal identified by SignalId to the location specified by SignalDataPtr and stores the length of the dynamical length signal at the position given by the Length parameter. The received meta data is stored at the position given by the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.15 Com_ReceiveSignalGroupWithMetaData

[SWS_Com_91007] Definition of API function Com_ReceiveSignalGroupWithMetaData

Service Name	Com_ReceiveSignalGroupWithMetaData	
Syntax	`uint8 Com_ReceiveSignalGroupWithMetaData ( Com_SignalGroupIdType SignalGroupId, uint8* MetaDataPtr )`	
Service ID [hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be received.
Parameters (inout)	None	
Parameters (out)	MetaDataPtr	Pointer to the meta data of the signal group

▽

△

Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_ReceiveSignalGroupWithMetaData copies the received signal group from the I-PDU to the shadow buffer. The received meta data is stored at the position given by the Meta DataPtr parameter.	
Available via	Com.h	

D.4.1.16 Com_ReceiveSignalGroupArrayWithMetaData

[SWS_Com_91008] Definition of API function Com_ReceiveSignalGroupArrayWithMetaData

Service Name	Com_ReceiveSignalGroupArrayWithMetaData	
Syntax	`uint8 Com_ReceiveSignalGroupArrayWithMetaData ( Com_SignalGroupIdType SignalGroupId, uint8* SignalGroupArrayPtr, uint8* MetaDataPtr )`	
Service ID [hex]	0x4c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be received.
Parameters (inout)	None	
Parameters (out)	SignalGroupArrayPtr	reference to the location where the received signal group array shall be stored
	MetaDataPtr	Pointer to the meta data of the signal group.
Return value	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description	The service Com_ReceiveSignalGroupArrayWithMetaData copies the received signal group array representation from the PDU to the location designated by SignalGroupAr-rayPtr. The received meta data is stored at the position given by the MetaDataPtr parameter.	
Available via	Com.h	

D.4.1.17 Com_InvalidateSignal

[SWS_Com_00203] Definition of API function Com_InvalidateSignal

Service Name	Com_InvalidateSignal
Syntax	`uint8 Com_InvalidateSignal ( Com_SignalIdType SignalId )`

▽

△

Service ID [hex]	0x10	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalId	Id of signal to be invalidated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	`E_OK`: service has been accepted `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group is stopped, no ComSignalDataInvalidValue is configured for the given signalId `COM_BUSY`: in case the TP-Buffer is locked
Description	The service Com_InvalidateSignal invalidates the signal with the given SignalId by setting its value to its configured ComSignalDataInvalidValue.	
Available via	Com.h	

D.4.1.18 Com_InvalidateSignalGroup

[SWS_Com_00557] Definition of API function Com_InvalidateSignalGroup

Service Name	Com_InvalidateSignalGroup	
Syntax	`uint8 Com_InvalidateSignalGroup ( Com_SignalGroupIdType SignalGroupId )`	
Service ID [hex]	0x1b	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SignalGroupId	Id of signal group to be invalidated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	`E_OK`: service has been accepted `COM_SERVICE_NOT_AVAILABLE`: corresponding I-PDU group is stopped, no ComSignalDataInvalidValue is configured for any of the group signals `COM_BUSY`: in case the TP-Buffer is locked
Description	The service Com_InvalidateSignalGroup invalidates all group signals of the signal group with the given SignalGroupId by setting their values to their configured ComSignalDataInvalidValues.	
Available via	Com.h	

D.4.1.19 ComUser_CbkRxAck

[SWS_Com_91019] Definition of configurable interface <ComUser_CbkRxAck>

Service Name	<ComUser_CbkRxAck>	

Syntax	`void <ComUser_CbkRxAck> (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant don't care	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id of the signal/signal group, which has been received.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback represents Notification Class 1 in Clause 3.6.2 (Notification classes) of [25]. It is called immediately after the message has been stored in the receiving message object.	
Available via	Configuration parameter ComUserHeaderInclude	

D.4.1.20 ComUser_CbkInv

[SWS_Com_91021] Definition of configurable interface <ComUser_CbkInv>

Service Name	<ComUser_CbkInv>	

Syntax	`void <ComUser_CbkInv> (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant don't care	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id of the invalid signal/signal group.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback function corresponds to [SWS_Com_00680]. It is called after reception of an invalid signal or signal group respectively.	
Available via	Configuration parameter ComUserHeaderInclude	

D.4.1.21 ComUser_CbkRxTOut

[SWS_Com_91020] Definition of configurable interface <ComUser_CbkRxTOut>

Service Name	<ComUser_CbkRxTOut>	
Syntax	`void <ComUser_CbkRxTOut> ( CbkHandleIdType ComUserCbkHandleId )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant don't care	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id of the signal/signal group, for which a reception deadline violation has occurred.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback corresponds to Notification Class 3 in Clause 3.6.2 (Notification classes) of [25]. It is called immediately after a message reception error has been detected by the deadline monitoring mechanism.	
Available via	Configuration parameter ComUserHeaderInclude	

D.4.1.22 ComUser_CbkTxAck

[SWS_Com_91016] Definition of configurable interface <ComUser_CbkTxAck>

Service Name	<ComUser_CbkTxAck>	
Syntax	`void <ComUser_CbkTxAck> ( CbkHandleIdType ComUserCbkHandleId )`	
Sync/Async	Synchronous	
Reentrancy	don't care	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id of the signal/signal group, which transmission is acknowledged
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback represents Notification Class 2 in Clause 3.6.2 (Notification classes) of [25]. It is called immediately after successful transmission of the I-PDU containing the message.	
Available via	Configuration parameter ComUserHeaderInclude	

D.4.1.23 ComUser_CbkTxErr

[SWS_Com_91017] Definition of configurable interface <ComUser_CbkTxErr>

Service Name	<ComUser_CbkTxErr>	

Syntax	`void <ComUser_CbkTxErr> (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Sync/Async	Synchronous	
Reentrancy	don't care	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id of the signal/signal group, which could not be transmitted.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback corresponds to Notification Class 4 in Clause 3.6.2 (Notification classes) of [25]. It is called in case the transmission is not possible because the corresponding I-PDU group is stopped.	
Available via	Configuration parameter ComUserHeaderInclude	

D.4.1.24 ComUser_CbkTxTOut

[SWS_Com_91018] Definition of configurable interface <ComUser_CbkTxTOut>

Service Name	<ComUser_CbkTxTOut>	

Syntax	`void <ComUser_CbkTxTOut> (`
	`  CbkHandleIdType ComUserCbkHandleId`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant don't care	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id of the signal/signal group, for which a transmission deadline violation has occurred.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This callback corresponds to Notification Class 4 in Clause 3.6.2 (Notification classes) of [25]. It is called immediately after a message transmission error has been detected by the deadline monitoring mechanism.	
Available via	Configuration parameter ComUserHeaderInclude	

D.4.2 APIs to handle ComIPdus

D.4.2.1 Com_Transmit

[SWS_Com_00931] Definition of API function Com_Transmit

Service Name	Com_Transmit	

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

Service ID [hex]	0x51	
Sync/Async	Synchronous	
Reentrancy	Reentrant Reentrant for different TxPduIds. Non reentrant for the same TxPduId.	
Parameters (in)	TxPduId	Identifier of the I-PDU to be transmitted.
	PduInfoPtr	Length of and pointer to the signal data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `COM_BUSY`: Transmit request has been rejected due to minimum delay time violation. `E_NOT_OK`: Transmit request has not been accepted.
Description	Requests transmission of an I-PDU with Large Data Path.	
Available via	Com.h	

D.4.2.2 ComUser_ComCbkTriggerTransmit

[SWS_Com_91029] Definition of configurable interface <ComUser_ComCbkTriggerTransmit>

Service Name	<ComUser_ComCbkTriggerTransmit>	

Syntax	`Std_ReturnType <ComUser_ComCbkTriggerTransmit> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  PduInfoType* PduInfoPtr`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the ID of the SDU that is requested to be transmitted
	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 (inout)	None	
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	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.3 ComUser_ComCbkRxIndication

[SWS_Com_91031] Definition of configurable interface <ComUser_ComCbkRx Indication>

Service Name	<ComUser_ComCbkRxIndication>	

Syntax	`void <ComUser_ComCbkRxIndication> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Sync/Async	Asynchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to received I-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 a lower layer communication interface module.	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.4 ComUser_ComCbkTxConfirmation

[SWS_Com_91030] Definition of configurable interface <ComUser_ComCbkTx Confirmation>

Service Name	<ComUser_ComCbkTxConfirmation>

Syntax	`void <ComUser_ComCbkTxConfirmation> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  Std_ReturnType result`
	`)`

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the PDU that has been transmitted
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.5 ComUser_ComCbkCopyRxData

[SWS_Com_91027] Definition of configurable interface <ComUser_ComCbkCopyRxData>

Service Name	<ComUser_ComCbkCopyRxData>	
Syntax	`BufReq_ReturnType <ComUser_ComCbkCopyRxData> ( CbkHandleIdType ComUserCbkHandleId, const PduInfoType* info, PduLengthType* bufferSizePtr )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the received I-PDU
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer after data has been copied.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data copied successfully `BUFREQ_E_NOT_OK`: Data was not copied because an error occurred.
Description	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.6 ComUser_ComCbkStartOfReception

[SWS_Com_91026] Definition of configurable interface <ComUser_ComCbkStartOfReception>

Service Name	<ComUser_ComCbkStartOfReception>	

Syntax	`BufReq_ReturnType <ComUser_ComCbkStartOfReception> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  const PduInfoType* info,`
	`  PduLengthType TpSduLength,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the I-PDU
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. `BUFREQ_E_NOT_OK`: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. `BUFREQ_E_OVFL`: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.7 ComUser_ComCbkTpRxIndication

[SWS_Com_91028] Definition of configurable interface <ComUser_ComCbkTpRxIndication>

Service Name	<ComUser_ComCbkTpRxIndication>	

Syntax	`void <ComUser_ComCbkTpRxIndication> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  Std_ReturnType result`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the received I-PDU

▽

△

	result	E_OK: The I-PDU was received successfully. E_NOT_OK: Reception of the I-PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called after an I-PDU has been received via the TP API, the result indicates whether the reception was successful or not.	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.8 ComUser_ComCbkTpTxConfirmation

[SWS_Com_91025] Definition of configurable interface <ComUser_ComCbkTpTxConfirmation>

Service Name	<ComUser_ComCbkTpTxConfirmation>	

Syntax	`void <ComUser_ComCbkTpTxConfirmation> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  Std_ReturnType result`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the transmitted I-PDU
	result	E_OK - transmission successful E_NOT_OK - transmission not successful
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called after a Signal has been transmitted via the TP-API on its network.	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.4.2.9 ComUser_ComCbkCopyTxData

[SWS_Com_91024] Definition of configurable interface <ComUser_ComCbkCopyTxData>

Service Name	<ComUser_ComCbkCopyTxData>	

Syntax	`BufReq_ReturnType <ComUser_ComCbkCopyTxData> (`
	`  CbkHandleIdType ComUserCbkHandleId,`
	`  const PduInfoType* info,`
	`  const RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Non Reentrant for same ComUserCbkHandleId, otherwise Reentrant	
Parameters (in)	ComUserCbkHandleId	Com user callback handle Id corresponding to the transmitted I-PDU
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the Sdu DataPtr may be a NULL_PTR.
	retry	Will not be handled by Com and its upper layer.
Parameters (inout)	None	
Parameters (out)	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value	BufReq_ReturnType	`BUFREQ_OK`: Data has been copied to the transmit buffer completely as requested. `BUFREQ_E_BUSY`: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. `BUFREQ_E_NOT_OK`: Data has not been copied. Request failed.
Description	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr	
Available via	ComUserHeaderInclude ([ECUC_Com_10032])	

D.5 Dcm

D.5.1 DataServices

D.5.1.1 ReadData

If `DcmDspDataUsePort` is set to `USE_DATA_SYNCH_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91090] Definition of configurable interface Xxx_ReadData

Service Name	Xxx_ReadData	
Syntax	`Std_ReturnType Xxx_ReadData ( uint8* Data, uint16 DataLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x68	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	DataLength	Length in byte of the data to be read
Parameters (inout)	None	
Parameters (out)	Data	Buffer where the requested data shall be copied to
	ErrorCode	If the operation Xxx_ReadData returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function requests to the application a data value of a DID/PID if DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY.	
Available via	Dcm_Externals.h	

If `DcmDspDataUsePort` is set to `USE_DATA_ASYNCH_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91091] Definition of configurable interface Xxx_ReadData

Service Name	Xxx_ReadData	
Syntax	`Std_ReturnType Xxx_ReadData ( Dcm_OpStatusType OpStatus, uint8* Data, uint16 DataLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x7d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	OpStatus	Status of the current operation
	DataLength	Length in byte of the data to be read
Parameters (inout)	None	
Parameters (out)	Data	Buffer where the requested data shall be copied to
	ErrorCode	If the operation Xxx_ReadData returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function requests to the application a data value of a DID/PID if DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_PROXY..	
Available via	Dcm_Externals.h	

D.5.1.2 WriteData

If DcmDspDataType is set to USE_DATA_SYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91095] Definition of configurable interface Xxx_WriteData

Service Name	Xxx_WriteData	
Syntax	`Std_ReturnType Xxx_WriteData ( const uint8* Data, uint16 DataLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x85	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Data	Buffer containing the data to be written
	DataLength	Length in byte of the data to be written
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description	This function requests the application to write a data value of a DID if DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY.	
Available via	Dcm_Externals.h	

If DcmDspDataType is set to USE_DATA_ASYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91092] Definition of configurable interface Xxx_WriteData

Service Name	Xxx_WriteData	
Syntax	`Std_ReturnType Xxx_WriteData ( const uint8* Data, uint16 DataLength, Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x91	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Data	Buffer containing the data to be written
	DataLength	Length in byte of the data to be written
	OpStatus	Status of the current operation
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_WriteData returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.



△

Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description	This function requests the application to write a data value of a DID if DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_PROXY.	
Available via	Dcm_Externals.h	

D.5.1.3 ConditionCheckRead

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_FNC, or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR, USE_DATA_ASYNCH_FNC_ERROR or USE_DATA_ASYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91011] Definition of configurable interface Xxx_ConditionCheck Read

Service Name	Xxx_ConditionCheckRead	
Syntax	`Std_ReturnType Xxx_ConditionCheckRead ( Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x37	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	OpStatus	Status of the current operation
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_ConditionCheckRead returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description	This function requests to the application if the conditions to read the Data are correct.	
Available via	Dcm_Externals.h	

D.5.1.4 ReadDataLength

If DcmDspDataUsePort is set to USE_DATA_SYNCH_CLIENT_SERVER, or USE_DATA_SYNCH_FNC, or USE_DATA_SYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91096] Definition of configurable interface Xxx_ReadDataLength

Service Name	Xxx_ReadDataLength	
Syntax	`Std_ReturnType Xxx_ReadDataLength ( uint16* DataLength )`	
Service ID [hex]	0x36	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	DataLength	Length in byte of the data to be read
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description	This function requests the application to return the data length in byte of a Data.	
Available via	Dcm_Externals.h	

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_CLIENT_SERVER or USE_DATA_ASYNCH_CLIENT_SERVER_ERROR or USE_DATA_ASYNCH_FNC or USE_DATA_ASYNCH_FNC_ERROR, or USE_DATA_SYNCH_FNC_PROXY, or USE_DATA_ASYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91097] Definition of configurable interface Xxx_ReadDataLength

Service Name	Xxx_ReadDataLength	
Syntax	`Std_ReturnType Xxx_ReadDataLength ( Dcm_OpStatusType OpStatus, uint16* DataLength )`	
Service ID [hex]	0x8c	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	OpStatus	Status of the current operation
Parameters (inout)	None	
Parameters (out)	DataLength	Length in byte of the data to be read
Return value	Std_ReturnType	E_OK: this value is always returned. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description	This function requests the application to return the data length in byte of a Data.	
Available via	Dcm_Externals.h	

D.5.1.5 GetScalingInformation

If DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91093] Definition of configurable interface Xxx_GetScalingInformation

Service Name	Xxx_GetScalingInformation	
Syntax	`Std_ReturnType Xxx_GetScalingInformation ( uint8* ScalingInfo, uint8 ScalingInfoLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x90	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== (USE_DATA_SYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataInfoRef->DcmDspDataScalingInfoSize)} != NULL)	
Parameters (in)	ScalingInfoLength	Length in byte of the scaling information to be read
Parameters (inout)	None	
Parameters (out)	ScalingInfo	Scaling information (scalingByte and scalingByteExtension)
	ErrorCode	If the operation Xxx_GetScalingInformation returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function requests to the application for the scaling information of a Data.	
Available via	Dcm_Externals.h	

If `DcmDspDataUsePort` is set to `USE_DATA_ASYNCH_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91094] Definition of configurable interface Xxx_GetScalingInformation

Service Name	Xxx_GetScalingInformation	
Syntax	`Std_ReturnType Xxx_GetScalingInformation ( Dcm_OpStatusType OpStatus, uint8* ScalingInfo, uint8 ScalingInfoLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x8f	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== (USE_DATA_ASYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataInfoRef->DcmDspDataScalingInfoSize)} != NULL)	
Parameters (in)	OpStatus	Status of the current operation
	ScalingInfoLength	Length in byte of the scaling information to be read
Parameters (inout)	None	
Parameters (out)	ScalingInfo	Scaling information (scalingByte and scalingByteExtension)
	ErrorCode	If the operation Xxx_GetScalingInformation returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.

▽

△

Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description	This function requests to the application for the scaling information of a Data.	
Available via	Dcm_Externals.h	

D.5.1.6 ReturnControlToECU

If DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY or USE_DATA_ASYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91098] Definition of configurable interface Xxx_ReturnControlToECU

Service Name	Xxx_ReturnControlToECU	

Syntax	`Std_ReturnType Xxx_ReturnControlToECU (`
	`  uint8* controlMask,`
	`  uint8 controlMaskLength,`
	`  Dcm_NegativeResponseCodeType ErrorCode`
	`)`	

Service ID [hex]	0x8e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== (USE_DATA_SYNCH_FNC_PROXY || USE_DATA_ASYNCH_FNC_PROXY)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl)} != NULL)	
Parameters (in)	controlMask	Control enable mask
	controlMaskLength	Control enable mask length in byte
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_ReturnControlToECU returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description	This function requests to the application to return control to ECU of an IOControl.	
Available via	Dcm_Externals.h	

D.5.1.7 ResetToDefault

If DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91099] Definition of configurable interface Xxx_ResetToDefault

Service Name	Xxx_ResetToDefault	
Syntax	`Std_ReturnType Xxx_ResetToDefault ( uint8* controlMask, uint8 controlMaskLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x8d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== USE_DATA_SYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE)	
Parameters (in)	controlMask	Control enable mask
	controlMaskLength	Control enable mask length in byte
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_ResetToDefault returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function requests to the application to reset an IOControl to default value.	
Available via	Dcm_Externals.h	

If `DcmDspDataUsePort` is set to `USE_DATA_ASYNCH_CLIENT_SERVER` or `USE_DATA_ASYNCH_FNC` or `USE_DATA_ASYNCH_CLIENT_SERVER_ERROR` or `USE_DATA_ASYNCH_FNC_ERROR`, the following definition is used:

[SWS_Dcm_01314] Definition of configurable interface Xxx_ResetToDefault

Service Name	Xxx_ResetToDefault	
Syntax	`Std_ReturnType Xxx_ResetToDefault ( Dcm_OpStatusType OpStatus, [Dcm_ControlMask_{DID}Type controlMask], [uint8* controlMask], Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x3c	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_FNC || USE_DATA_ASYNCH_FNC_ERROR|| USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR||)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidResetToDefault)} == TRUE)	
Parameters (in)	OpStatus	Status of the current operation
	controlMask	–
	Type	Dcm_ControlMask_{DID}Type

▽

△

	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} <= 0x04)
	controlMask	–
	Type	uint8*
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef -> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL)&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} => 0x05)
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_ResetToDefault returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description	This function requests to the application to reset an IOControl to default value.	
Available via	Dcm_Externals.h	

Note: Square brackets [] indicate that an argument is optional.

D.5.1.8 FreezeCurrentState

If DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91101] Definition of configurable interface Xxx_FreezeCurrentState

Service Name	Xxx_FreezeCurrentState	
Syntax	`Std_ReturnType Xxx_FreezeCurrentState ( uint8* controlMask, uint8 controlMaskLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x77	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== USE_DATA_SYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE)	
Parameters (in)	controlMask	Control enable mask
	controlMaskLength	Control enable mask length in byte

▽

△

Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_FreezeCurrentState returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description	This function requests to the application to freeze the current state of an IOControl.	
Available via	Dcm_Externals.h	

If DcmDspDataUsePort is set to USE_DATA_ASYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91102] Definition of configurable interface Xxx_FreezeCurrentState

Service Name	Xxx_FreezeCurrentState	
Syntax	`Std_ReturnType Xxx_FreezeCurrentState ( Dcm_OpStatusType OpStatus, uint8* controlMask, uint8 controlMaskLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x88	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== USE_DATA_ASYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidFreezeCurrentState)} == TRUE)	
Parameters (in)	OpStatus	Status of the current operation
	controlMask	Control enable mask
	controlMaskLength	Control enable mask length in byte
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_FreezeCurrentState returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish.
Description	This function requests to the application to freeze the current state of an IOControl.	
Available via	Dcm_Externals.h	

D.5.1.9 ShortTermAdjustment

If DcmDspDataUsePort is set to USE_DATA_SYNCH_FNC_PROXY, the following definition is used:

[SWS_Dcm_91103] Definition of configurable interface Xxx_ShortTermAdjustment

Service Name	Xxx_ShortTermAdjustment	
Syntax	`Std_ReturnType Xxx_ShortTermAdjustment ( const uint8* ControlStateInfo, uint16 DataLength, uint8* controlMask, uint8 controlMaskLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x87	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== USE_DATA_SYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE)	
Parameters (in)	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	DataLength	ControlState information length in byte
	controlMask	Control enable mask
	controlMaskLength	Control enable mask length in byte
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_ShortTermAdjustment returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function requests to the application to adjust the IO signal.	
Available via	Dcm_Externals.h	

If `DcmDspDataUsePort` is set to `USE_DATA_ASYNCH_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91104] Definition of configurable interface Xxx_ShortTermAdjustment

Service Name	Xxx_ShortTermAdjustment
Syntax	`Std_ReturnType Xxx_ShortTermAdjustment ( const uint8* ControlStateInfo, uint16 DataLength, Dcm_OpStatusType OpStatus, uint8* controlMask, uint8 controlMaskLength, Dcm_NegativeResponseCodeType* ErrorCode )`
Service ID [hex]	0x83
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)}== USE_DATA_ASYNCH_FNC_PROXY) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == TRUE)

▽

△

Parameters (in)	ControlStateInfo	ControlState information contained in the ControlOptionRecord parameter of the InputOutputControlByIdentifier diagnostic request
	DataLength	ControlState information length in byte
	OpStatus	Status of the current operation
	controlMask	Control enable mask
	controlMaskLength	Control enable mask length in byte
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_ShortTermAdjustment returns value E_NOT_OK, the DCM module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful `DCM_E_PENDING`: Request is not yet finished. Further call(s) required to finish.
Description	This function requests to the application to adjust the IO signal.	
Available via	Dcm_Externals.h	

D.5.2 RoutineServices

D.5.2.1 Xxx_Start Operation

[SWS_Dcm_91105] Definition of configurable interface Xxx_Start

Service Name	Xxx_Start	

Syntax	`Std_ReturnType Xxx_Start (`
	`  Dcm_OpStatusType OpStatus,`
	`  uint8* dataInOut,`
	`  uint16* currentDataLength,`
	`  Dcm_NegativeResponseCodeType* ErrorCode`
	`)`	

Service ID [hex]	0x80	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	OpStatus	Status of the current operation
Parameters (inout)	dataInOut	Input and output data in the routine control request / response
	currentDataLength	This parameter contains the length in bytes of the dataInOut array. It include fixed length and variable length data.
Parameters (out)	ErrorCode	If the operation Xxx_Start returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful. `DCM_E_PENDING`: Request is not yet finished. Further call(s) required to finish. `DCM_E_FORCE_RCRRP`: application requests the transmission of a response Pending (NRC 0x78)
Description	This function requests to the application to start the execution of a routine.	

▽

△

Available via	Dcm_Externals.h

D.5.2.2 Xxx_StartConfirmation Operation

If `DcmDspRoutineFncSignature` is set to `ROUTINE_FNC_NORMAL` or `ROUTINE_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91016] Definition of configurable interface Xxx_StartConfirmation

Service Name	Xxx_StartConfirmation	
Syntax	`Std_ReturnType Xxx_StartConfirmation ( Dcm_ConfirmationStatusType ConfirmationStatus )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfirmationStatus	Confirmation status of a StartRoutine request
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function indicates the transmission of a response to a StartRoutine request	
Available via	Dcm_Externals.h	

D.5.2.3 Xxx_Stop Operation

If `DcmDspRoutineFncSignature` is set to `ROUTINE_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91106] Definition of configurable interface Xxx_Stop

Service Name	Xxx_Stop	
Syntax	`Std_ReturnType Xxx_Stop ( Dcm_OpStatusType OpStatus, uint8* dataInOut, uint16* currentDataLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x81	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	OpStatus	Status of the current operation
Parameters (inout)	dataInOut	Input and output data in the routine control request / response

▽

△

	currentDataLength	This parameter contains the length in bytes of the dataInOut array. It include fixed length and variable length data.
Parameters (out)	ErrorCode	If the operation Xxx_Stop returns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value.
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful. DCM_E_PENDING: Request is not yet finished. Further call(s) required to finish DCM_E_FORCE_RCRRP: application requests the transmission of a response Pending (NRC 0x78)
Description	This function requests to the application to stop the execution of a routine	
Available via	Dcm_Externals.h	

D.5.2.4 Xxx_StopConfirmation Operation

If DcmDspRoutineFncSignature is set to ROUTINE_FNC_NORMAL or ROUTINE_FNC_PROXY, the following definition is used:

[SWS_Dcm_91017] Definition of configurable interface Xxx_StopConfirmation

Service Name	Xxx_StopConfirmation	
Syntax	Std_ReturnType Xxx_StopConfirmation (Dcm_ConfirmationStatusType ConfirmationStatus)	
Service ID [hex]	0x69	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfirmationStatus	Dcm_ConfirmationStatus Confirmation status of a StopRoutine request
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request was successful. E_NOT_OK: Request was not successful.
Description	This function indicates the transmission of a response to a StopRoutine request	
Available via	Dcm_Externals.h	

D.5.2.5 Xxx_RequestResults Operation

If DcmDspRoutineFncSignature is set to ROUTINE_FNC_PROXY, the following definition is used:

[SWS_Dcm_91107] Definition of configurable interface Xxx_RequestResults

Service Name	Xxx_RequestResults	
Syntax	`Std_ReturnType Xxx_RequestResults ( Dcm_OpStatusType OpStatus, uint8* dataInOut, uint16* currentDataLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x82	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	OpStatus	Status of the current operation
Parameters (inout)	dataInOut	Input and output data in the routine control request / response If variable length routine input data is used, this parameter contains the length in bytes of the dataInVar array. If variable length routine output data is used, this parameter contains the length in bytes of the dataOutVar parameter.
	currentDataLength	This parameter contains the length in bytes of the dataInOut array. It include fixed length and variable length data.
Parameters (out)	ErrorCode	If the operation Xxx_RequestResults returns value E_NOT_OK, the Dcm module shall send a negative response with N
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful. `DCM_E_PENDING`: Request is not yet finished. Further call(s) required to finish `DCM_E_FORCE_RCRRP`: application requests the transmission of a response Pending (NRC 0x78)
Description	This function requests to the application the result of a routine execution	
Available via	Dcm_Externals.h	

D.5.2.6 Xxx_RequestResultsConfirmation Operation

If `DcmDspRoutineFncSignature` is set to `ROUTINE_FNC_NORMAL` or `ROUTINE_FNC_PROXY`, the following definition is used:

[SWS_Dcm_91018] Definition of configurable interface Xxx_RequestResultsConfirmation

Service Name	Xxx_RequestResultsConfirmation	
Syntax	`Std_ReturnType Xxx_RequestResultsConfirmation ( Dcm_ConfirmationStatusType ConfirmationStatus )`	
Service ID [hex]	0x70	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfirmationStatus	Confirmation status of a RequestRoutineResults request
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.

▽

△

Description	This function indicates the transmission of a response to a RequestRoutineResults request
Available via	Dcm_Externals.h

D.5.3 ServiceRequestNotification

From the point of view of the DCM, the operations has the following signatures:

D.5.3.1 Indication

[SWS_Dcm_01341] Definition of configurable interface Xxx_Indication

Service Name	Xxx_Indication	

Syntax	`Std_ReturnType Xxx_Indication (`
	`  uint8 SID,`
	`  const uint8* RequestData,`
	`  uint32 DataSize,`
	`  uint8 ReqType,`
	`  uint16 ConnectionId,`
	`  Dcm_NegativeResponseCodeType* ErrorCode,`
	`  Dcm_ProtocolType ProtocolType,`
	`  uint16 TesterSourceAddress`
	`)`	

Service ID [hex]	0x65	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SID	–
	RequestData	Complete request data (diagnostic buffer), except the service ID
	DataSize	Number of valid bytes in the RequestData parameter
	ReqType	Addressing type of the request(0=physical request,1=functional request)
	ConnectionId	Unique connection identifier
	ProtocolType	Type of the protocol to be indicated
	TesterSourceAddress	source address of the tester
Parameters (inout)	None	
Parameters (out)	ErrorCode	If the operation Xxx_Indication re- turns value E_NOT_OK, the Dcm module shall send a negative response with NRC code equal to the parameter ErrorCode parameter value
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful. `E_REQUEST_NOT_ACCEPTED` : Request not accepted
Description	This function indicates to the application that a service is about to be executed and allows the application to reject the execution of the service request	
Available via	Dcm_Externals.h	

D.5.3.2 Confirmation

[SWS_Dcm_01342] Definition of configurable interface Xxx_Confirmation

Service Name	Xxx_Confirmation	
Syntax	`Std_ReturnType Xxx_Confirmation ( uint8 SID, uint8 ReqType, uint16 ConnectionId, Dcm_ConfirmationStatusType ConfirmationStatus, Dcm_ProtocolType ProtocolType, uint16 TesterSourceAddress )`	
Service ID [hex]	0x66	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SID	Value of service identifier
	ReqType	Addressing type of the request(0=physical request,1=functional request)
	ConnectionId	Unique connection identifier
	ConfirmationStatus	Confirmation of a successful transmission or a transmission error of a diagnostic service.
	ProtocolType	Type of Dcm Protocol
	TesterSourceAddress	source address of the tester
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request was successful. `E_NOT_OK`: Request was not successful.
Description	This function confirms to the application the successful transmission or a transmission error of a diagnostic service.	
Available via	Dcm_Externals.h	

D.6 Dem

D.6.1 Dem_ClearPrestoredFreezeFrame

[SWS_Dem_00193] Definition of API function Dem_ClearPrestoredFreezeFrame

Service Name	Dem_ClearPrestoredFreezeFrame	
Syntax	`Std_ReturnType Dem_ClearPrestoredFreezeFrame ( Dem_EventIdType EventId )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: Clear prestored freeze frame was successful E_NOT_OK: Clear prestored freeze frame failed
Description	Clears a prestored freeze frame of a specific event. This API can only be used through the RTE and therefore no declaration is exported via Dem.h. API Availability: This API will be available only if ({ecuc(Dem/DemConfigSet/DemEvent Parameter.DemFFPrestorageSupported)} == true)	
Available via	Dem.h	

D.6.2 Dem_PrestoreFreezeFrame

[SWS_Dem_00188] Definition of API function Dem_PrestoreFreezeFrame

Service Name	Dem_PrestoreFreezeFrame	
Syntax	`Std_ReturnType Dem_PrestoreFreezeFrame ( Dem_EventIdType EventId )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK Freeze frame prestorage was successful E_NOT_OK Freeze frame prestorage failed
Description	Captures the freeze frame data for a specific event. This API can only be used through the RTE and therefore no declaration is exported via Dem.h. API Availability: This API will be available only if ({ecuc(Dem/DemConfigSet/DemEvent Parameter.DemFFPrestorageSupported)} == true)	
Available via	Dem.h	

D.6.3 Dem_ResetEventDebounceStatus

[SWS_Dem_00683] Definition of API function Dem_ResetEventDebounceStatus

Service Name	Dem_ResetEventDebounceStatus	
Syntax	`Std_ReturnType Dem_ResetEventDebounceStatus ( Dem_EventIdType EventId, Dem_DebounceResetStatusType DebounceResetStatus )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId.

▽

△

	DebounceResetStatus	Freeze or reset the internal debounce counter/timer of the specified event.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Operation was successful `E_NOT_OK`: Only on development error
Description	Control the internal debounce counter/timer by BSW modules and SW-Cs. The event qualification will not be affected by these debounce state changes. This API is available for BSW modules as soon as Dem_PreInit has been completed (refer to [SWS_Dem_00438] and [SWS_Dem_00167]).	
Available via	Dem.h	

D.6.4 Dem_ResetEventStatus

[SWS_Dem_00185] Definition of API function Dem_ResetEventStatus

Service Name	Dem_ResetEventStatus	
Syntax	`Std_ReturnType Dem_ResetEventStatus ( Dem_EventIdType EventId )`	
Service ID [hex]	0x05	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request to reset the event status was successful accepted. `E_NOT_OK`: Request to reset the event status failed or is not allowed, because the event is already tested in this operation cycle.
Description	Resets the event failed status. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.	
Available via	Dem.h	

D.6.5 Dem_SetEventDisabled

[SWS_Dem_00312] Definition of API function Dem_SetEventDisabled

Service Name	Dem_SetEventDisabled
Syntax	`Std_ReturnType Dem_SetEventDisabled ( Dem_EventIdType EventId )`
Service ID [hex]	0x51
Sync/Async	Synchronous
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.

▽

△

Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK` set of event to disabled was successfull. `E_NOT_OK` set of event disabled failed
Description	Service for reporting the event as disabled to the Dem for the PID $41 computation. API is needed in OBD-relevant ECUs only. API Availability: This API will be available only if ({ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT)	
Available via	Dem.h	

D.6.6 Dem_SetEventStatus

[SWS_Dem_00183] Definition of API function Dem_SetEventStatus

Service Name	Dem_SetEventStatus	
Syntax	`Std_ReturnType Dem_SetEventStatus ( Dem_EventIdType EventId, Dem_EventStatusType EventStatus )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId.
	EventStatus	Monitor test result
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: set of event status was successful `E_NOT_OK`: Event status setting or processing failed or could not be accepted.
Description	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/DemConfigSet/DemEventParameter/DemEventReportingType} == STANDARD_REPORTING)	
Available via	Dem.h	

D.6.7 Dem_GetDTCOfEvent

[SWS_Dem_00198] Definition of API function Dem_GetDTCOfEvent

Service Name	Dem_GetDTCOfEvent	
Syntax	`Std_ReturnType Dem_GetDTCOfEvent ( Dem_EventIdType EventId, Dem_DTCFormatType DTCFormat, uint32* DTCOfEvent )`	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
	DTCFormat	Defines the output-format of the requested DTC value.
Parameters (inout)	None	
Parameters (out)	DTCOfEvent	Receives the DTC value in respective format returned by this function. If the return value of the function is other than E_OK this parameter does not contain valid data.
Return value	Std_ReturnType	`E_OK`: get of DTC was successful `E_NOT_OK`: the call was not successful `DEM_E_NO_DTC_AVAILABLE`: there is no DTC configured in the requested format
Description	Gets the DTC of an event.	
Available via	Dem.h	

D.6.8 Dem_GetDebouncingOfEvent

[SWS_Dem_00730] Definition of API function Dem_GetDebouncingOfEvent

Service Name	Dem_GetDebouncingOfEvent	
Syntax	`Std_ReturnType Dem_GetDebouncingOfEvent ( Dem_EventIdType EventId, Dem_DebouncingStateType* DebouncingState )`	
Service ID [hex]	0x9f	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	DebouncingState	Bit 0 Temporarily Defective (corresponds to 0 < FDC < 127) Bit 1 finally Defective (corresponds to FDC = 127) Bit 2 temporarily healed (corresponds to -128 < FDC < 0) Bit 3 Test complete (corresponds to FDC = -128 or FDC = 127) Bit 4 DTR Update (= Test complete && Debouncing complete && enable conditions / storage conditions fulfilled)
Return value	Std_ReturnType	`E_OK`: get of debouncing status per event state successful `E_NOT_OK`: get of debouncing per event state failed
Description	Gets the debouncing status of an event. This function shall not be used for EventId with native debouncing within their functions. It is rather for EventIds using debouncing within the Dem.	
Available via	Dem.h	

D.6.9 Dem_GetEventAvailable

[SWS_Dem_91040] Definition of API function Dem_GetEventAvailable

Service Name	Dem_GetEventAvailable	
Syntax	`Std_ReturnType Dem_GetEventAvailable ( Dem_EventIdType EventId, boolean* AvailableStatus )`	
Service ID [hex]	0xbe	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	AvailableStatus	TRUE if the event is available. FALSE if the event is not available.
Return value	Std_ReturnType	`E_OK` : Event availability has been obtained. `E_NOT_OK` : Event availability cannot be obtained.
Description	Get the Event's availability.	
Available via	Dem.h	

D.6.10 Dem_GetEventExtendedDataRecordEx

[SWS_Dem_01190] Definition of API function Dem_GetEventExtendedDataRecordEx

Service Name	Dem_GetEventExtendedDataRecordEx	
Syntax	`Std_ReturnType Dem_GetEventExtendedDataRecordEx ( Dem_EventIdType EventId, uint8 RecordNumber, uint8* DestBuffer, uint16* BufSize )`	
Service ID [hex]	0x6d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
	RecordNumber	Identification of requested Extended data record. Valid values are between 0x01 and 0xEF as defined in ISO14229-1.
Parameters (inout)	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out)	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the extended data shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value	Std_ReturnType	`E_OK`: Operation was successful `E_NOT_OK`: Operation could not be performed `DEM_NO_SUCH_ELEMENT`: The requested event data is not currently stored (but the request was valid) OR the requested record number is not supported by the event. `DEM_BUFFER_TOO_SMALL`: The provided buffer size is too small.

▽

△

Description	Gets the data of an extended data record by event.
Available via	Dem.h

D.6.11 Dem_GetEventFreezeFrameDataEx

[SWS_Dem_01191] Definition of API function Dem_GetEventFreezeFrameDataEx

Service Name	Dem_GetEventFreezeFrameDataEx	

Syntax	`Std_ReturnType Dem_GetEventFreezeFrameDataEx (`
	`  Dem_EventIdType EventId,`
	`  uint8 RecordNumber,`
	`  uint16 DataId,`
	`  uint8* DestBuffer,`
	`  uint16* BufSize`
	`)`	

Service ID [hex]	0x6e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
	RecordNumber	This parameter is a unique identifier for a freeze frame record as defined in ISO14229-1. **0xFF means most recent freeze frame record is returned.** 0x00 is only supported if the Dem module supports WWH-OBD (refer to DemOBDSupport)
	DataId	This parameter specifies the DID (ISO14229-1) that shall be copied to the destination buffer.
Parameters (inout)	BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Parameters (out)	DestBuffer	This parameter contains a byte pointer that points to the buffer, to which the freeze frame data record shall be written to. The format is raw hexadecimal values and contains no header-information.
Return value	Std_ReturnType	`E_OK`: Operation was successful `E_NOT_OK`: Operation could not be performed `DEM_NO_SUCH_ELEMENT`: The requested event data is not currently stored (but the request was valid) OR The requested record number is not supported by the event OR The requested DID is not supported by the freeze frame. `DEM_BUFFER_TOO_SMALL`: The provided buffer size is too small.
Description	Gets the data of a freeze frame by event.	
Available via	Dem.h	

D.6.12 Dem_GetEventUdsStatus

[SWS_Dem_91008] Definition of API function Dem_GetEventUdsStatus

Service Name	Dem_GetEventUdsStatus	
Syntax	`Std_ReturnType Dem_GetEventUdsStatus ( Dem_EventIdType EventId, Dem_UdsStatusByteType* UDSStatusByte )`	
Service ID [hex]	0xb6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	UDSStatusByte	UDS DTC status byte of the requested event (refer to chapter "Status bit support"). If the return value of the function call is E_NOT_OK, this parameter does not contain valid data.
Return value	Std_ReturnType	`E_OK`: get of event status was successful `E_NOT_OK`: get of event status failed
Description	Gets the current UDS status byte assigned to the DTC for the event	
Available via	Dem.h	

D.6.13 Dem_GetFaultDetectionCounter

[SWS_Dem_00203] Definition of API function Dem_GetFaultDetectionCounter

Service Name	Dem_GetFaultDetectionCounter	
Syntax	`Std_ReturnType Dem_GetFaultDetectionCounter ( Dem_EventIdType EventId, sint8* FaultDetectionCounter )`	
Service ID [hex]	0x3e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	FaultDetectionCounter	This parameter receives the Fault Detection Counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1
Return value	Std_ReturnType	`E_OK`: request was successful `E_NOT_OK`: request failed `DEM_E_NO_FDC_AVAILABLE`: there is no fault detection counter available for the requested event
Description	Gets the fault detection counter of an event. This API can only be used through the RTE, and therefore no declaration is exported via Dem.h.	
Available via	Dem.h	

D.6.14 Dem_GetMonitorStatus

[SWS_Dem_91007] Definition of API function Dem_GetMonitorStatus

Service Name	Dem_GetMonitorStatus	
Syntax	`Std_ReturnType Dem_GetMonitorStatus ( Dem_EventIdType EventID, Dem_MonitorStatusType* MonitorStatus )`	
Service ID [hex]	0xb5	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventID	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	MonitorStatus	Monitor status byte of the requested event. If the return value of the function call is E_NOT_OK, this parameter does not contain valid data.
Return value	Std_ReturnType	`E_OK`: Get monitor status was successful, `E_NOT_OK`: getting the monitor status failed.
Description	Gets the current monitor status for an event.	
Available via	Dem.h	

D.6.15 Dem_SetEventStatusWithMonitorData

[SWS_Dem_91037] Definition of API function Dem_SetEventStatusWithMonitor Data

Service Name	Dem_SetEventStatusWithMonitorData	
Syntax	`Std_ReturnType Dem_SetEventStatusWithMonitorData ( Dem_EventIdType EventId, Dem_EventStatusType EventStatus, Dem_MonitorDataType monitorData0, Dem_MonitorDataType monitorData1 )`	
Service ID [hex]	0xbd	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId.
	EventStatus	Monitor test result
	monitorData0	–
	monitorData1	–
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: set of event status was successful `E_NOT_OK`: Event status setting or processing failed or could not be accepted.
Description	This API will be available only if ({Dem/DemConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING_WITH_MONITOR_DATA)	
Available via	Dem.h	

D.6.16 Dem_SetWIRStatus

[SWS_Dem_00839] Definition of API function Dem_SetWIRStatus

Service Name	Dem_SetWIRStatus	
Syntax	`Std_ReturnType Dem_SetWIRStatus ( Dem_EventIdType EventId, boolean WIRStatus )`	
Service ID [hex]	0x7a	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different EventIds. Non reentrant for the same EventId.	
Parameters (in)	EventId	Identification of an event by assigned EventId. The Event Number is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.:Result of configuration of Event Numbers in DEM (Max is either 255 or 65535)
	WIRStatus	Requested status of event related WIR-bit (regarding to the current status of function inhibition) WIRStatus = TRUE -> WIR-bit shall be set to "1" WIRStatus = FALSE -> WIR-bit shall be set to "0"
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request to set the WIR status was successful. `E_NOT_OK`: Request to set the WIR status was not accepted (e.g. disabled controlDTCSetting) and should be repeated.
Description	Sets the WIR status bit via failsafe SW-Cs. This API can only be used through the RTE and therefore no declaration is exported via Dem.h.	
Available via	Dem.h	

D.6.17 <Module>_DemGetFaultDetectionCounter<ForEvent>

[SWS_Dem_00263] Definition of configurable interface <Module>_DemGetFault DetectionCounter<ForEvent>

Service Name	<Module>_DemGetFaultDetectionCounter<ForEvent>	
Syntax	`Std_ReturnType <Module>_DemGetFaultDetectionCounter<ForEvent> ( sint8* FaultDetectionCounter )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	FaultDetectionCounter	This parameter receives the fault detection counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1
Return value	Std_ReturnType	`E_OK`: request was successful `E_NOT_OK`: request failed

▽

△

Description	Gets the current fault detection counter value. There is one c-callback per event using monitor-internal debouncing, if no port interface is provided by the Dem.
Available via	Dem_Externals.h

D.6.18 <Module>_DemInitMonitorFor<EventName>

[SWS_Dem_00256] Definition of configurable interface <Module>_DemInitMonitorFor<EventName>

Service Name	<Module>_DemInitMonitorFor<EventName>	
Syntax	`Std_ReturnType <Module>_DemInitMonitorFor<EventName> ( Dem_InitMonitorReasonType InitMonitorReason )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	InitMonitorReason	Specific (re-)initialization reason evaluated from the monitor to identify the initialization kind to be performed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	Return value unused - only for compatibility with according RTE operation.
Description	Inits the diagnostic monitor of a specific event. There is one separate callback per event (if configured), if no port interface is provided by the Dem.	
Available via	Dem_Externals.h	

D.6.19 <Module>_DemTriggerOnMonitorStatus

[SWS_Dem_91010] Definition of configurable interface <Module>_DemTriggerOnMonitorStatus

Service Name	<Module>_DemTriggerOnMonitorStatus	
Syntax	`Std_ReturnType <Module>_DemTriggerOnMonitorStatus ( void )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: this function always returns `E_OK` for compatibility reasons with RTE

▽

△

Description	Triggers on changes of the monitor status. Called synchronously in context of event status reporting.
Available via	Dem_Externals.h

D.6.20 <Module>_DemClearEventAllowed<ForCondition>

[SWS_Dem_00563] Definition of configurable interface <Module>_DemClear EventAllowed<ForCondition>

Service Name	<Module>_DemClearEventAllowed<ForCondition>	
Syntax	`Std_ReturnType <Module>_DemClearEventAllowed<ForCondition> ( boolean* Allowed )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	Allowed	True - clearance of event is allowed False - clearance of event is not allowed
Return value	Std_ReturnType	`E_OK`: Operation was successful `E_NOT_OK`: Operation failed
Description	Triggers on DTC-deletion, which is not allowed if the out-parameter returns False. There is one separate callback per condition, which can be assigned to one or several events, if no port interface is provided by the Dem. Parameter "Allowed" will be unchanged in case E_NOT_OK is returned.	
Available via	Dem_Externals.h	

D.6.21 <Module>_DemTriggerOnEventData

[SWS_Dem_00562] Definition of configurable interface <Module>_DemTriggerOn EventData

Service Name	<Module>_DemTriggerOnEventData	
Syntax	`Std_ReturnType <Module>_DemTriggerOnEventData ( Dem_EventIdType EventId )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventId	Identification of an event by assigned EventId.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	`E_OK`: this function always returns `E_OK` for compatibility reasons with RTE.
Description	Triggers on changes of the event related data in the event memory.	
Available via	Dem_Externals.h	

D.6.22 <Module>_DemTriggerOnEventUdsStatus

[SWS_Dem_91006] Definition of configurable interface <Module>_DemTriggerOnEventUdsStatus

Service Name	<Module>_DemTriggerOnEventUdsStatus	
Syntax	`Std_ReturnType <Module>_DemTriggerOnEventUdsStatus ( Dem_UdsStatusByteType EventStatusByteOld, Dem_UdsStatusByteType EventStatusByteNew )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	EventStatusByteOld	UDS DTC status byte of event before change (refer to chapter "Status bit support").
	EventStatusByteNew	UDS DTC status byte of event after change (refer to chapter "Status bit support").
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: this function always returns `E_OK` for compatibility reasons with RTE
Description	Triggers on changes of the UDS DTC status byte.	
Available via	Dem_Externals.h	

D.7 FiM

D.7.1 FiM_GetFunctionPermission

[SWS_Fim_00011] Definition of API function FiM_GetFunctionPermission

Service Name	FiM_GetFunctionPermission
Syntax	`Std_ReturnType FiM_GetFunctionPermission ( FiM_FunctionIdType FID, boolean* Permission )`
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	FID	Identification of a functionality by assigned FID. The FunctionId is configured in the FIM. Min.: 1 (0: Indication of no functionality) Max.: Result of configuration of FIDs in FIM (Max is either 255 or 65535)
Parameters (inout)	None	
Parameters (out)	Permission	TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
Return value	Std_ReturnType	`E_OK`: The request is accepted `E_NOT_OK`: The request is not accepted, ie. initialization of FIM not completed
Description	This service reports the permission state to the functionality.	
Available via	FiM.h	

D.7.2 FiM_SetFunctionAvailable

[SWS_Fim_00106] Definition of API function FiM_SetFunctionAvailable

Service Name	FiM_SetFunctionAvailable	
Syntax	`Std_ReturnType FiM_SetFunctionAvailable ( FiM_FunctionIdType FID, boolean Availability )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FID	Identification of a functionality by assigned FID.
	Availability	The permission of the requested FID: TRUE: Function is available. FALSE: Function is not available.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request is accepted `E_NOT_OK`: Request is not accepted (e.g. invalid FID is given)
Description	This service sets the availability of a function. The function is only available if FiMAvailability Support is configured as True.	
Available via	FiM.h	

E Referenced Service Interfaces

E.1 Os

OS Service interfaces are not provided by SwCluC.

E.2 NvM

E.2.1 NvM_BlockIdType

[SWS_NvM_00471] Definition of ImplementationDataType NvM_BlockIdType

Name	NvM_BlockIdType		
Kind	Type		
Derived from	uint16		
Range	0..2^(16- NvMDataset SelectionBits)-1	–	--
Description	Identification of a NVRAM block via a unique block identifier. Reserved NVRAM block IDs: 0 -> to derive multi block request results via NvM_GetErrorStatus 1 -> redundant NVRAM block which holds the configuration ID		
Variation	–		
Available via	Rte_NvM_Type.h		

E.2.2 NvM_BlockRequestType

[SWS_NvM_91002] Definition of ImplementationDataType NvM_BlockRequestType

Name	NvM_BlockRequestType		
Kind	Type		
Derived from	uint8		
Range	NVM_READ_BLOCK	0x00	NvM_ReadBlock/ NvM_Read PRAMBlock was performed on the block
	NVM_WRITE_BLOCK	0x01	NvM_WriteBlock/ NvM_Write PRAMBlock was performed on the block
	NVM_RESTORE_BLOCK_ DEFAULTS	0x02	NvM_RestoreBlockDefaults/ Nv M_RestorePRAMBlockDefaults was performed on the block
	NVM_ERASE_NV_BLOCK	0x03	NvM_EraseNvBlock was performed on the block
	NVM_INVALIDATE_NV_ BLOCK	0x04	NvM_InvalidateNvBlock was performed on the block
	NVM_READ_ALL_BLOCK	0x05	NvM_ReadAll has finished processing this block

▽

△

Description	Identifies the type of request performed on a block when signaled via the callback function
Variation	–
Available via	Rte_NvM_Type.h

E.2.3 NvM_InitBlockRequestType

[SWS_NvM_91123] Definition of ImplementationDataType NvM_InitBlockRequestType

Name	NvM_InitBlockRequestType		
Kind	Type		
Derived from	uint8		
Range	NVM_INIT_READ_BLOCK	0x00	NvM_ReadBlock/ NvM_ReadPRAMBlock is requested on the block
	NVM_INIT_RESTORE_BLOCK_DEFAULTS	0x01	NvM_RestoreBlockDefaults/ NvM_RestorePRAMBlockDefaults is requested on the block
	NVM_INIT_READ_ALL_BLOCK	0x02	NvM_ReadAll is processing this block
	NVM_INIT_FIRST_INIT_ALL	0x03	NvM_FirstInitAll is processing this block
Description	Identifies the type of request performed on a block when signaled via the callback function		
Variation	–		
Available via	Rte_NvM_Type.h		

E.2.4 NvM_RequestResultType

[SWS_NvM_00470] Definition of ImplementationDataType NvM_RequestResultType

Name	NvM_RequestResultType		
Kind	Type		
Derived from	uint8		
Range	NVM_REQ_OK	0x00	The last asynchronous request has been finished successfully. This shall be the default value after reset. This status shall have the value 0.
	NVM_REQ_NOT_OK	0x01	The last asynchronous read/write/control request has been finished unsuccessfully.
	NVM_REQ_PENDING	0x02	An asynchronous read/write/control request is currently pending.

▽

△

	NVM_REQ_INTEGRITY_FAILED	0x03	The result of the last asynchronous request NvM_ReadBlock or NvM_ReadAll is a data integrity failure. Note: In case of NvM_ReadBlock the content of the RAM block has changed but has become invalid. The application is responsible to renew and validate the RAM block content.
	NVM_REQ_BLOCK_SKIPPED	0x04	The referenced block was skipped during execution of NvM_ReadAll or NvM_WriteAll, e.g. Dataset NVRAM blocks (NvM_ReadAll) or NVRAM blocks without a permanently configured RAM block.
	NVM_REQ_NV_INVALIDATED	0x05	The referenced NV block is invalidated.
	NVM_REQ_CANCELED	0x06	The multi block request NvM_WriteAll was canceled by calling NvM_CancelWriteAll. Or Any single block job request (NvM_ReadBlock, NvM_WriteBlock, NvM_EraseNvBlock, NvM_InvalidateNvBlock and NvM_RestoreBlockDefaults) was canceled by calling NvM_CancelJobs.
	NVM_REQ_RESTORED_DEFAULTS	0x08	The referenced NV block had the default values copied to the RAM image.
Description	This is an asynchronous request result returned by the API service NvM_GetErrorStatus. The availability of an asynchronous request result can be additionally signaled via a callback function.		
Variation	–		
Available via	Rte_NvM_Type.h		

E.2.5 NvMService

[SWS_NvM_00734] Definition of ClientServerInterface NvMService

Name	NvMService		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	EraseBlock
Comment	Service to erase a NV block.
Relates to	NvM_EraseNvBlock

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	GetDataIndex	
Comment	Service for getting the currently set DataIndex of a dataset NVRAM block	
Relates to	NvM_GetDataIndex	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`WHERE`
	`  isConfigClass2;`	

Parameters	DataIndex	
	Type	uint8
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetErrorStatus	
Comment	Service to read the block dependent error/status information.	
Relates to	NvM_GetErrorStatus	
Variation	–	
Parameters	RequestResult	
	Type	NvM_RequestResultType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	InvalidateNvBlock
Comment	Service to invalidate a NV block.
Relates to	NvM_InvalidateNvBlock

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	ReadBlock	
Comment	Service to copy the data of the NV block to its corresponding RAM block.	
Relates to	NvM_ReadBlock	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Parameters	DstPtr	
	Type	VoidPtr
	Direction	IN
	Comment	The parameter "DstPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	ReadPRAMBlock	
Comment	Service to copy the data of the NV block to its corresponding RAM block.	
Relates to	NvM_ReadPRAMBlock	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Possible Errors	E_OK E_NOT_OK	

Operation	RestoreBlockDefaults	
Comment	Service to restore the default data to its corresponding RAM block.	
Relates to	NvM_RestoreBlockDefaults	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Parameters	DstPtr	
	Type	VoidPtr
	Direction	IN

▽

△

	Comment	The parameter "DstPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	RestorePRAMBlockDefaults
Comment	Service to restore the default data to its corresponding RAM block.
Relates to	NvM_RestorePRAMBlockDefaults

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	SetDataIndex	
Comment	Service for setting the DataIndex of a dataset NVRAM block.	
Relates to	NvM_SetDataIndex	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3`
	`  = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`  blockMgmTypes`
	`  = ECV.subEltList("NvM/NvMBlockDescriptor/NvMBlockManagementType");`
	`  isMgd(mgmtType)`
	`  = mgmtType.value() == "NVM_BLOCK_DATASET";`
	`  datasetMgdCount`
	`  = blockMgmTypes.filter(isMgd).count();`
	`WHERE`
	`  (isConfigClass2 OR isConfigClass3) AND (datasetMgdCount GT 0);`	

Parameters	DataIndex	
	Type	uint8
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetRamBlockStatus
Comment	Service for setting the RAM block status of an NVRAM block.
Relates to	NvM_SetRamBlockStatus

▽

△

Variation	**LET**
	  nvmBlockUseSetRamBlockStatus
	  = ECV.*subEltList*("NvM/NvMBlockDescriptor/
	  NvMBlockUseSetRamBlockStatus");
	  useSetRamBlockStatus(useApi)
	  = useApi.*value*() == true;
	  useSetRamBlockStatusCount
	  = nvmBlockUseSetRamBlockStatus.*filter*(useSetRamBlockStatus).count(
	  );
	WHERE
	  (useSetRamBlockStatusCount **GT** 0);

Parameters	BlockChanged
	Type boolean
	Direction IN
	Comment –
	Variation –
Possible Errors	E_OK E_NOT_OK

Operation	WriteBlock
Comment	Service to copy the data of the RAM block to its corresponding NV block.
Relates to	NvM_WriteBlock

Variation	**FOR**
	  configClass : ECV.*subEltList*("NvM/NvMCommon/NvMApiConfigClass");
	LET
	  isConfigClass2
	  = configClass.*value*() == "NVM_API_CONFIG_CLASS_2";
	  isConfigClass3
	  = configClass.*value*() == "NVM_API_CONFIG_CLASS_3";
	WHERE
	  isConfigClass2 **OR** isConfigClass3;

Parameters	SrcPtr
	Type ConstVoidPtr
	Direction IN
	Comment The parameter "SrcPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation –
Possible Errors	E_OK E_NOT_OK

Operation	WritePRAMBlock
Comment	Service to copy the data of the RAM block to its corresponding NV block.
Relates to	NvM_WritePRAMBlock

Variation	**FOR**
	  configClass : ECV.*subEltList*("NvM/NvMCommon/NvMApiConfigClass");
	LET
	  isConfigClass2
	  = configClass.*value*() == "NVM_API_CONFIG_CLASS_2";
	  isConfigClass3
	  = configClass.*value*() == "NVM_API_CONFIG_CLASS_3";
	WHERE
	  isConfigClass2 **OR** isConfigClass3;

Possible Errors	E_OK E_NOT_OK

E.2.6 NvMAdmin

[SWS_NvM_00737] Definition of ClientServerInterface NvMAdmin

Name	NvMAdmin		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetBlockProtection	
Comment	Service for setting/resetting the write protection for a NV block.	
Relates to	NvM_SetBlockProtection	

Variation	**FOR**
	  configClass : ECV.*subEltList*("NvM/NvMCommon/NvMApiConfigClass");
	LET
	  isConfigClass3 = configClass.*value*() == "NVM_API_CONFIG_CLASS_3";
	WHERE
	  isConfigClass3;	

Parameters	ProtectionEnabled	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.2.7 NvMNotifyJobFinished

[SWS_NvM_00735] Definition of ClientServerInterface NvMNotifyJobFinished

Name	NvMNotifyJobFinished		
Comment	Callback that is called when a job has finished		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful

Operation	JobFinished	
Comment	Callback that gets called if a job has finished	
Relates to	–	
Variation	–	
Parameters	BlockRequest	
	Type	NvM_BlockRequestType
	Direction	IN
	Comment	–
	Variation	–

▽

△

	JobResult	
	Type	NvM_RequestResultType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK	

E.2.8 NvMNotifyInitBlock

[SWS_NvM_00736] Definition of ClientServerInterface NvMNotifyInitBlock

Name	NvMNotifyInitBlock		
Comment	Callback that is called by the NvM module when default data needs to be restored to the RAM image		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	RAM block content was updated
	1	RTE_E_RAM_ UNCHANGED	RAM block content was not changed

Operation	InitBlock	
Comment	This callback is called if the initialization of a block has completed.	
Relates to	–	
Variation	–	
Parameters	InitBlockRequest	
	Type	NvM_InitBlockRequestType
	Direction	IN
	Comment	--
	Variation	–
Possible Errors	–	

E.2.9 NvMMirror

[SWS_NvM_00738] Definition of ClientServerInterface NvMMirror

Name	NvMMirror		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	ReadRamBlockFromNvM	
Comment	Block specific callback routine which shall be called in order to let the application copy data from NvM module's mirror to RAM block.	
Relates to	NvM_ReadRamBlockFromNvm	
Variation	–	
Parameters	SrcPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	The parameter "SrcPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	WriteRamBlockToNvM	
Comment	Block specific callback routine which shall be called in order to let the application copy data from RAM block to NvM module's mirror.	
Relates to	NvM_WriteRamBlockToNvm	
Variation	–	
Parameters	DstPtr	
	Type	VoidPtr
	Direction	IN
	Comment	The parameter "DstPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.2.10 PS_{Block}

[SWS_NvM_00847] Definition of Port PS_{Block} provided by module NvM

Name	PS_{Block}		
Kind	ProvidedPort	***Interface***	NvMService
Description	–		
Port Defined Argument Value(s)	***Type***	NvM_BlockIdType	
	Value	{BlockId}	
Variation			

```
FOR
  nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor");
LET
  Block
    = nvBlockDescriptor.shortname();
  BlockId
    =  nvBlockDescriptor.subElt("NvMNvramBlockIdentifier").value();
  UsePort
    = nvBlockDescriptor.subElt("NvMBlockUsePort").value() == true;
WHERE
  UsePort;
```

E.2.11 PAdmin_{Block}

[SWS_NvM_00843] Definition of Port PAdmin_{Block} provided by module NvM

Name	PAdmin_{Block}		
Kind	ProvidedPort	***Interface***	NvMAdmin
Description	–		
Port Defined Argument Value(s)	***Type***	NvM_BlockIdType	
	Value	{BlockId}	

Variation	**FOR**
	  nvBlockDescriptor : ECV.*subEltList*("NvM/NvMBlockDescriptor");
	LET
	  Block
	  = nvBlockDescriptor.*shortname*();
	  BlockId
	  = nvBlockDescriptor.*subElt*("NvMNvramBlockIdentifier").*value*();
	  UsePort
	  = nvBlockDescriptor.*subElt*("NvMBlockUsePort").*value*() == true;
	WHERE
	  UsePort;		

E.2.12 PNJF_{Block}

[SWS_NvM_00846] Definition of Port PNJF_{Block} required by module NvM

Name	PNJF_{Block}		
Kind	RequiredPort	***Interface***	NvMNotifyJobFinished
Description	–		

Variation	**FOR**
	  nvBlockDescriptor : ECV.*subEltList*("NvM/NvMBlockDescriptor");
	LET
	  Block
	  = nvBlockDescriptor.*shortname*();
	  UsePort
	  = nvBlockDescriptor.*subElt*("NvMBlockUsePort").*value*() == true;
	  SingleBlockCallbackDef
	  = nvBlockDescriptor.*subElt*("NvMSingleBlockCallback").isDefined();
	  SingleBlockCallbackFncDef
	  = nvBlockDescriptor.*subElt*("NvMSingleBlockCallback/
	  NvMSingleBlockCallbackFnc").isDefined();
	WHERE
	  UsePort **AND** SingleBlockCallbackDef **AND NOT** SingleBlockCallbackFncDef;		

E.2.13 PNIB_{Block}

[SWS_NvM_00845] Definition of Port PNIB_{Block} required by module NvM

Name	PNIB_{Block}		
Kind	RequiredPort	***Interface***	NvMNotifyInitBlock
Description	–		

Variation	**FOR**
	  nvBlockDescriptor : ECV.*subEltList*("NvM/NvMBlockDescriptor");
	LET
	  Block
	  = nvBlockDescriptor.*shortname*();
	  UsePort
	  = nvBlockDescriptor.*subElt*("NvMBlockUsePort").*value*() == true;
	  InitBlockCallbackDef
	  = nvBlockDescriptor.*subElt*("NvMInitBlockCallback").isDefined();
	  InitBlockCallbackFncDef
	  = nvBlockDescriptor.*subElt*("NvMInitBlockCallback/
	  NvMInitBlockCallbackFnc").isDefined();
	WHERE
	  UsePort **AND** InitBlockCallbackDef **AND NOT** InitBlockCallbackFncDef;		

E.2.14 PM_{Block}

[SWS_NvM_00844] Definition of Port PM_{Block} required by module NvM

Name	PM_{Block}		
Kind	RequiredPort	***Interface***	NvMMirror
Description	–		

Variation	**FOR**
	  nvBlockDescriptor : ECV.*subEltList*("NvM/NvMBlockDescriptor");
	LET
	  Block
	  = nvBlockDescriptor.*shortname*();
	  UsePort
	  = nvBlockDescriptor.*subElt*("NvMBlockUsePort").*value*() == true;
	  UsePortSyncMech
	  = nvBlockDescriptor.*subElt*("NvMBlockUseSyncMechanism").*value*() ==
	  true;
	WHERE
	  UsePort **AND** UsePortSyncMech;		

E.3 Dcm

E.3.1 Implementation Data Types

E.3.1.1 Dcm_OpStatusType

[SWS_Dcm_00984] Definition of ImplementationDataType Dcm_OpStatusType

Name	Dcm_OpStatusType		
Kind	Type		
Derived from	uint8		
Range	DCM_INITIAL	0x00	Indicates the initial call to the operation
	DCM_PENDING	0x01	Indicates that a pending return has been done on the previous call of the operation
	DCM_CANCEL	0x02	Indicates that the DCM requests to cancel the pending operation
	DCM_FORCE_RCRRP_OK	0x03	Confirm a response pending transmission
Description	–		
Variation	–		
Available via	Rte_Dcm_Type.h		

E.3.1.2 Dcm_ConfirmationStatusType

[SWS_Dcm_00983] Definition of ImplementationDataType Dcm_ConfirmationStatusType

Name	Dcm_ConfirmationStatusType		
Kind	Type		
Derived from	uint8		
Range	DCM_RES_POS_OK	0x00	–
	DCM_RES_POS_NOT_OK	0x01	–
	DCM_RES_NEG_OK	0x02	–
	DCM_RES_NEG_NOT_OK	0x03	–
Description	–		
Variation	–		
Available via	Rte_Dcm_Type.h		

[SWS_Dcm_00979] Definition of ImplementationDataType Dcm_ProtocolType

Name	Dcm_ProtocolType
Kind	Type
Derived from	uint8

▽

△

Range	DCM_OBD_ON_CAN	0x00	OBD on CAN (ISO15765-4; ISO15031-5)
	DCM_OBD_ON_FLEXRAY	0x01	(OBD on Flexray (Manufacturer specific; ISO15031-5))
	DCM_OBD_ON_IP	0x02	(OBD on Internet Protocol (Manufacturer specific; ISO15031-5))
	DCM_UDS_ON_CAN	0x03	UDS on CAN (ISO15765-3; ISO14229-1)
	DCM_UDS_ON_FLEXRAY	0x04	UDS on FlexRay (Manufacturer specific; ISO14229-1)
	DCM_UDS_ON_IP	0x05	(UDS on Internet Protocol (Manufacturer specific; ISO14229-1))
	DCM_ROE_ON_CAN	0x06	Response On Event on CAN
	DCM_ROE_ON_FLEXRAY	0x07	Response On Event on FlexRay
	DCM_ROE_ON_IP	0x08	(Response on Event on Internet Protocol)
	DCM_PERIODICTRANS_ON_CAN	0x09	Periodic Transmission on CAN
	DCM_PERIODICTRANS_ON_FLEXRAY	0x0A	Periodic Transmission on FlexRay
	DCM_PERIODICTRANS_ON_IP	0x0B	(Periodic Transmission on Internet Protocol)
	DCM_NO_ACTIVE_PROTOCOL	0x0C	No protocol has been started
	DCM_UDS_ON_LIN	0x0D	UDS on LIN (ISO14229-1; ISO14229-7)
	Reserved for further AUTOSAR implementation	0x0E..0xEF	–
	DCM_SUPPLIER_1	0xF0	Reserved for SW supplier specific.
	DCM_SUPPLIER_2	0xF1	Reserved for SW supplier specific.
	DCM_SUPPLIER_3	0xF2	Reserved for SW supplier specific.
	DCM_SUPPLIER_4	0xF3	Reserved for SW supplier specific.
	DCM_SUPPLIER_5	0xF4	Reserved for SW supplier specific.
	DCM_SUPPLIER_6	0xF5	Reserved for SW supplier specific.
	DCM_SUPPLIER_7	0xF6	Reserved for SW supplier specific.
	DCM_SUPPLIER_8	0xF7	Reserved for SW supplier specific.
	DCM_SUPPLIER_9	0xF8	Reserved for SW supplier specific.
	DCM_SUPPLIER_10	0xF9	Reserved for SW supplier specific.
	DCM_SUPPLIER_11	0xFA	Reserved for SW supplier specific.

▽

△

	DCM_SUPPLIER_12	0xFB	Reserved for SW supplier specific.
	DCM_SUPPLIER_13	0xFC	Reserved for SW supplier specific.
	DCM_SUPPLIER_14	0xFD	Reserved for SW supplier specific.
	DCM_SUPPLIER_15	0xFE	Reserved for SW supplier specific.
Description	Protocol type definition		
Variation	–		
Available via	Rte_Dcm_Type.h		

E.3.1.3 Dcm_NegativeResponseCodeType

[SWS_Dcm_00980] Definition of ImplementationDataType Dcm_NegativeResponseCodeType

Name	Dcm_NegativeResponseCodeType		
Kind	Type		
Derived from	uint8		
Range	DCM_POS_RESP	0x00	PR
	range of values 0x01..0x0F reserved by ISO 14229	0x01..0x0F	ISOSAERESRVD
	DCM_E_ GENERALREJECT	0x10	GR
	DCM_E_ SERVICENOTSUPPORTED	0x11	SNS
	DCM_E_SUBFUNCTION-NOTSUPPORTED	0x12	SFNS
	DCM_E_INCORRECTMES-SAGELENGTHORINVALID-FORMAT	0x13	IMLOIF
	DCM_E_ RESPONSETOOLONG	0x14	RTL
	range of values 0x15..0x20 reserved by ISO 14229	0x15..0x20	ISOSAERESRVD
	DCM_E_ BUSYREPEATREQUEST	0x21	BRR
	DCM_E_CONDITIONSNOT-CORRECT	0x22	CNC
	value 0x23 reserved by ISO 14229	0x23	ISOSAERESRVD
	DCM_E_REQUESTSE-QUENCEERROR	0x24	RSE
	DCM_E_ NORESPONSEFROMSUB-NETCOMPONENT	0x25	NRFSC

▽

△

	DCM_E_FAILUREPREVENTSEXECUTIONOFREQUESTEDACTION	0x26	FPEORA
	range of values 0x27..0x30 reserved by ISO 14229	0x27..0x30	ISOSAERESRVD
	DCM_E_REQUESTOUTOFRANGE	0x31	ROOR
	value 0x32 reserved by ISO 14229	0x32	ISOSAERESRVD
	DCM_E_SECURITYACCESSDENIED	0x33	SAD
	DCM_E_AUTHENTICATIONREQUIRED	0x34	AR
	DCM_E_INVALIDKEY	0x35	IK
	DCM_E_EXCEEDNUMBEROFATTEMPTS	0x36	ENOA
	DCM_E_REQUIREDTIMEDELAYNOTEXPIRED	0x37	RTDNE
	DCM_E_SECUREDATATRANSMISSIONREQUIRED	0x38	SDTR
	DCM_E_SECUREDATATRANSMISSIONNOTALLOWED	0x39	SDTNA
	DCM_E_SECUREDATAVERIFICATIONFAILED	0x3A	SDTF
	range of values 0x3B..0x4F reserved by ISO 15764	0x3B..0x4F	RBEDLSD
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDTIMEPERIOD	0x50	CVFITP
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDSIGNATURE	0x51	CVFIS
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDCHAINOFTRUST	0x52	CVFICOT
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDTYPE	0x53	CVFIT
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDFORMAT	0x54	CVFIF
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDCONTENT	0x55	CVFIC
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDSCOPE	0x56	CVFIS
	DCM_E_CERTIFICATEVERIFICATIONFAILEDINVALIDCERTIFICATEREVOKED	0x57	CVFIC
	DCM_E_OWNERSHIPVERIFICATIONFAILED	0x58	OVF

▽

△

	DCM_E_CHALLENGECAL-CULATIONFAILED	0x59	CCF
	DCM_E_SETTINGAC-CESSRIGHTSFAILED	0x5A	SARF
	DCM_E_ SESSIONKEYCREATION-DERIVATIONFAILED	0x5B	SKCDF
	DCM_E_CONFIGURA-TIONDATAUSAGEFAILED	0x5C	CDUF
	DCM_E_DEAUTHENTICA-TIONFAILED	0x5D	DAF
	range of values 0x5E..0x6F reserved by ISO 14229	0x5E..0x6F	ISOSAERESRVD
	DCM_E_UPLOADDOWN-LOADNOTACCEPTED	0x70	UDNA
	DCM_E_TRANSFERDATA-SUSPENDED	0x71	TDS
	DCM_E_GENERALPRO-GRAMMINGFAILURE	0x72	GPF
	DCM_E_WRONGBLOCK-SEQUENCECOUNTER	0x73	WBSC
	range of values 0x74..0x77 reserved by ISO 14229	0x74..0x77	ISOSAERESRVD
	range of values 0x79..0x7D reserved by ISO 14229	0x79..0x7D	ISOSAERESRVD
	DCM_E_SUBFUNCTION-NOTSUPPORTEDINAC-TIVESESSION	0x7E	SFNSIAS
	DCM_E_ SERVICENOTSUPPORTE-DINACTIVESESSION	0x7F	SNSIAS
	value 0x80 reserved by ISO 14229	0x80	ISOSAERESRVD
	DCM_E_RPMTOOHIGH	0x81	RPMTH
	DCM_E_RPMTOOLOW	0x82	RPMTL
	DCM_E_ ENGINEISRUNNING	0x83	EIR
	DCM_E_ ENGINEISNOTRUNNING	0x84	EINR
	DCM_E_ENGINERUN-TIMETOOLOW	0x85	ERTTL
	DCM_E_ TEMPERATURETOOHIGH	0x86	TEMPTH
	DCM_E_ TEMPERATURETOOLOW	0x87	TEMPTL
	DCM_E_ VEHICLESPEEDTOOHIGH	0x88	VSTH
	DCM_E_ VEHICLESPEEDTOOLOW	0x89	VSTL
	DCM_E_THROTTLE_ PEDALTOOHIGH	0x8A	TPTH
	DCM_E_THROTTLE_ PEDALTOOLOW	0x8B	TPTL

▽

△

	DCM_E_TRANSMISSION-RANGENOTINNEUTRAL	0x8C	TRNIN
	DCM_E_TRANSMISSION-RANGENOTINGEAR	0x8D	TRNIG
	value 0x8E reserved by ISO 14229	0x8E	ISOSAERESRVD
	DCM_E_BRAKESWITCH_ NOTCLOSED	0x8F	BSNC
	DCM_E_SHIFTERLEVER-NOTINPARK	0x90	SLNIP
	DCM_E_TORQUECON-VERTERCLUTCHLOCKED	0x91	TCCL
	DCM_E_ VOLTAGETOOHIGH	0x92	VTH
	DCM_E_ VOLTAGETOOLOW	0x93	VTL
	DCM_E_RESOURCETEM-PORARILYNOTAVAILABLE	0x94	RTNA
	range of values 0x95..0xEF reserved by ISO 14229	0x95..0xEF	RFSCNC
	DCM_E_VMSCNC_0	0xF0	VMSCNC
	DCM_E_VMSCNC_1	0xF1	VMSCNC1
	DCM_E_VMSCNC_2	0xF2	VMSCNC2
	DCM_E_VMSCNC_3	0xF3	VMSCNC3
	DCM_E_VMSCNC_4	0xF4	VMSCNC4
	DCM_E_VMSCNC_5	0xF5	VMSCNC5
	DCM_E_VMSCNC_6	0xF6	VMSCNC6
	DCM_E_VMSCNC_7	0xF7	VMSCNC7
	DCM_E_VMSCNC_8	0xF8	VMSCNC8
	DCM_E_VMSCNC_9	0xF9	VMSCNC9
	DCM_E_VMSCNC_A	0xFA	VMSCNCA
	DCM_E_VMSCNC_B	0xFB	VMSCNCB
	DCM_E_VMSCNC_C	0xFC	VMSCNCC
	DCM_E_VMSCNC_D	0xFD	VMSCNCD
	DCM_E_VMSCNC_E	0xFE	VMSCNCE
	value 0xFF reserved by ISO 14229	0xFF	ISOSAERESRVD
Description	This Table of available Negative Response Codes represents the allowed Response Codes an AUTOSAR SW Component shall return after a function call. For the allowed NRC of the executed Service ID please refer to the specification of the service in ISO14229-1 (UDS) and ISO15031-5 (OBD/CARB) (Response code parameter definition Table 18 of ISO 14229-1:2020).		
Variation	–		
Available via	Rte_Dcm_Type.h		

E.3.1.4 Dcm_DataElementType_{Data}Type

[SWS_Dcm_91051] Definition of ImplementationDataType Dcm_DataElement_{Data}Type

Name	Dcm_DataElement_{Data}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_DataElement_{Data}_ArrayType	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == ([S|U]INT[8|16|32|64]|FLOAT|DOUBLE)_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == ([S|U]INT[8|16|32|64]|FLOAT|DOUBLE)_N))
	Dcm_DataElement_{Data}_PrimitiveType	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == (BOOLEAN|[S|U]INT[8|16|32|64]|FLOAT|DOUBLE)) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == (BOOLEAN|[S|U]INT[8|16|32|64]|FLOAT|DOUBLE)))
	Dcm_DataElement_{Data}_StructuredType	DcmDspDidSignal with at least one DcmDspDidSignalCompositeRef.
Description	Common description for S/R and C/S data elements.	
Variation	(({Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort == USE_DATA_ELEMENT_SPECIFIC_INTERFACES}) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER))) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})	
Available via	Rte_Dcm_Type.h	

E.3.1.5 Dcm_DataElementType_{Data}ArrayType

[SWS_Dcm_91063] Definition of ImplementationDataType Dcm_DataElement_{Data}_ArrayType

Name	Dcm_DataElement_{Data}_ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == FLOAT_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == FLOAT_N))

▽

△

	float64	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == DOUBLE_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == DOUBLE_N))
	sint16	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == SINT16_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == SINT16_N))
	sint32	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == SINT32_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == SINT32_N))
	sint64	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT64_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT64_N))
	sint8	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == SINT8_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == SINT8_N))
	uint16	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == UINT16_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == UINT16_N))
	uint32	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == UINT32_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == UINT32_N))
	uint64	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT64_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT64_N))
	uint8	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == UINT8_N) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == UINT8_N))
Size	–	
Description	–	
Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == ([S|U]INT[8|16|32|64]|FLOAT|DOUBLE)_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == ([S|U]INT[8|16|32|64]|FLOAT|DOUBLE)_N)) && ({Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort == USE_DATA_ELEMENT_SPECIFIC_INTERFACES}) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER))) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})	
Available via	Rte_Dcm_Type.h	

E.3.1.6 Dcm_DataElementType_{Data}PrimitiveType

[SWS_Dcm_91062] Definition of ImplementationDataType Dcm_DataElement_{Data}_PrimitiveType

Name	Dcm_DataElement_{Data}_PrimitiveType	
Kind	Type	
Derived from	***Basetype***	***Variation***
	boolean	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == BOOLEAN) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == BOOLEAN))
	float32	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == FLOAT) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == FLOAT))
	float64	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == DOUBLE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == DOUBLE))
	sint16	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == SINT16) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == SINT16))
	sint32	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == SINT32) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == SINT32))
	sint64	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == SINT64) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == SINT64))
	sint8	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == SINT8) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == SINT8))
	uint16	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == UINT16) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == UINT16))
	uint32	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == UINT32) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == UINT32))
	uint64	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT64) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT64))
	uint8	(({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspData.?DcmDspDataType)} == UINT8) || ({ecuc(Dcm/?DcmConfigSet/?DcmDsp/?DcmDspPid/?DcmDspPidData/?DcmDspPidService01.?DcmDspPidDataType)} == UINT8))
Description	–	

▽

△

Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == (BOOLEAN|[S|U]INT[8|16|32|64]|FLOAT|DOUBLE)) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == (BOOLEAN|[S|U]INT[8|16|32|64]|FLOAT|DOUBLE))) && ({Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort == USE_DATA_ELEMENT_SPECIFIC_INTERFACES}) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER))) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})
Available via	Rte_Dcm_Type.h

E.3.1.7 Dcm_DataArrayTypeUint8_{Data}Type

[SWS_Dcm_01121] Definition of ImplementationDataType Dcm_DataArrayTypeUint8_{Data}Type

Name	Dcm_DataArrayTypeUint8_{Data}Type		
Kind	Array	***Element type***	uint8
Size	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataByteSize)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.DcmDspPidDataByteSize)})) Elements		
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_DATA_ELEMENT_SPECIFIC_INTERFACES) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT8_N) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataType)} == UINT8_DYN) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01.DcmDspPidDataType)} == UINT8_N)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})		
Available via	Rte_Dcm_Type.h		

E.3.1.8 {DID}_Struct_DataType

This data type has a different modeling as other data types. The {DID}_Struct_DataType datatype is modeled as prosa text only. At the time this specification was created there are no means to visualize this datatype with existing AUTOSAR tooling as table as all the other data types. Still AUTOSAR allows modeling such data types. Simply that they cannot be shown here as table.

[SWS_Dcm_91056] Definition of ImplementationDataType {DID}_Struct_DataType

Name	{DID}_Struct_DataType
Kind	Structure
Description	The elements of this structure data type is a composition of all DcmDspDataElement of the Dcm DspDid. Example: A DID with the 3 data elements uint32 data1, sint8 data2 and sint16 data 3, has a structure definition of struct { uint32 data1, sint8 data2, sint16 data 3}.
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == ((USE_ATOMIC_SENDER_RECEIVER_INTERFACE|| USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE) || USE_ATOMIC_NV_DATA_INTERFACE)) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})
Available via	Rte_Dcm_Type.h

E.3.1.9 Dcm_RangeArray_{Range}Type

[SWS_Dcm_01012] Definition of ImplementationDataType Dcm_RangeArray_{Range}Type

Name	Dcm_RangeArray_{Range}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRangeMaxDataLength)} Elements		
Description	–		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange.DcmDspDidRangeUsePort)} == TRUE Range = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidRange.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.10 Dcm_ScalingInfoArray_{Data}Type

[SWS_Dcm_01017] Definition of ImplementationDataType Dcm_ScalingInfoArray_{Data}Type

Name	Dcm_ScalingInfoArray_{Data}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDataInfo.DcmDspDataScalingInfoSize)} Elements		
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_SYNCH_CLIENT_SERVER) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData->DcmDspDataInfoRef.DcmDspDataScalingInfoSize)} != NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.11 Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01018] Definition of ImplementationDataType Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType

Name	Dcm_RequestDataOut_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	***Basetype***	***Variation***
	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == FLOAT
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == DOUBLE
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT32
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT64
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT32
	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT64
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT8
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32|64]|FLOAT|DOUBLE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.12 Dcm_RequestDataIn_{Routine}_{Signal}PrimitiveType

[SWS_Dcm_91054] Definition of ImplementationDataType Dcm_RequestDataIn_{Routine}_{Signal}PrimitiveType

Name	Dcm_RequestDataIn_{Routine}_{Signal}PrimitiveType	
Kind	Type	
Derived from	***Basetype***	***Variation***
	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == FLOAT
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == DOUBLE
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT32
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT64
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT32
	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT64
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT8
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32|64]|FLOAT|DOUBLE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.13 Dcm_RequestDataOut_{Routine}_{Signal}Type

[SWS_Dcm_91040] Definition of ImplementationDataType Dcm_RequestDataOut_{Routine}_{Signal}Type

Name	Dcm_RequestDataOut_{Routine}_{Signal}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_RequestDataOut_{Routine}_{Signal}PrimitiveType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64|FLOAT|DOUBLE)])
	Dcm_RequestDataOut_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N)
	Dcm_RequestDataOut_{Routine}_{Signal}StructuredType	DcmDspRequestRoutineResultsOutSignal with at least one DcmDspRequestRoutineResultsOutSignalCompositeSignalRef.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.14 Dcm_RequestDataIn_{Routine}_{Signal}Type

[SWS_Dcm_91052] Definition of ImplementationDataType Dcm_RequestDataIn_{Routine}_{Signal}Type

Name	Dcm_RequestDataIn_{Routine}_{Signal}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_RequestDataIn_{Routine}_{Signal}PrimitiveType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)])
	Dcm_RequestDataIn_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N)
	Dcm_RequestDataIn_{Routine}_{Signal}StructuredType	DcmDspRequestRoutineResultsInSignal with at least one DcmDspRequestRoutineResultsInSignalCompositeSignalRef.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.15 Dcm_RequestDataOut_{Routine}_{Signal}ArrayType

[SWS_Dcm_91041] Definition of ImplementationDataType Dcm_RequestDataOut_{Routine}_{Signal}ArrayType

Name	Dcm_RequestDataOut_{Routine}_{Signal}ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == FLOAT_N)
	float64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == DOUBLE_N)
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT64_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT64_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineParameterSize)} Elements	
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.16 Dcm_RequestDataIn_{Routine}_{Signal}ArrayType

[SWS_Dcm_91055] Definition of ImplementationDataType Dcm_RequestDataIn_{Routine}_{Signal}ArrayType

Name	Dcm_RequestDataIn_{Routine}_{Signal}ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == FLOAT_N
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == DOUBLE_N
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT16_N
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT32_N
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT64_N
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == SINT8_N
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT16_N
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT32_N
	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT64_N
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == UINT8_N
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineParameterSize)} Elements	
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.17 Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01019] Definition of ImplementationDataType Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type

Name	Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineParameterSize} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.18 Dcm_RequestFlexibleInArrayData_{Routine}_{Signal}Type

[SWS_Dcm_91053] Definition of ImplementationDataType Dcm_RequestFlexibleInArrayData_{Routine}_{Signal}Type

Name	Dcm_RequestFlexibleInArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineParameterSize} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.19 Dcm_StartDataIn_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01020] Definition of ImplementationDataType Dcm_StartDataIn_{Routine}_{Signal}PrimitivType

Name	Dcm_StartDataIn_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	***Basetype***	***Variation***
	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == FLOAT
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == DOUBLE
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT32
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT64
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT32
	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT64
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT8
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32|64]|FLOAT|DOUBLE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.20 Dcm_StartDataIn_{Routine}_{Signal}Type

[SWS_Dcm_91042] Definition of ImplementationDataType Dcm_StartDataIn_{Routine}_{Signal}Type

Name	Dcm_StartDataIn_{Routine}_{Signal}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_StartDataIn_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE))
	Dcm_StartDataIn_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N)
	Dcm_StartDataIn_{Routine}_{Signal}StructuredType	DcmDspStartRoutineInSignal with at least one DcmDspStartRoutineInSignalCompositeSignalRef.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.21 Dcm_StartDataIn_{Routine}_{Signal}ArrayType

[SWS_Dcm_91043] Definition of ImplementationDataType Dcm_StartDataIn_{Routine}_{Signal}ArrayType

Name	Dcm_StartDataIn_{Routine}_{Signal}ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == FLOAT_N
	float64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == DOUBLE_N)
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT64_N)

▽

△

	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT64_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineParameterSize)} Elements	
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.22 Dcm_StartDataOut_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01021] Definition of ImplementationDataType Dcm_StartDataOut_{Routine}_{Signal}PrimitivType

Name	Dcm_StartDataOut_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	***Basetype***	***Variation***
	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == FLOAT
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == DOUBLE
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT32
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT64

▽

△

	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT32
	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT64
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT8
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == [U|S]INT[8|16|32|64]|FLOAT|DOUBLE) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.23 Dcm_StartDataOut_{Routine}_{Signal}Type

[SWS_Dcm_91044] Definition of ImplementationDataType Dcm_StartDataOut_{Routine}_{Signal}Type

Name	Dcm_StartDataOut_{Routine}_{Signal}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_StartDataOut_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE))
	Dcm_StartDataOut_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N)
	Dcm_StartDataOut_{Routine}_{Signal}StructuredType	DcmDspStartRoutineOutSignal with at least one DcmDspStartRoutineOutSignalCompositeSignalRef.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.24 Dcm_StartDataOut_{Routine}_{Signal}ArrayType

[SWS_Dcm_91045] Definition of ImplementationDataType Dcm_StartDataOut_{Routine}_{Signal}ArrayType

Name	Dcm_StartDataOut_{Routine}_{Signal}ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == FLOAT_N
	float64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == DOUBLE_N)
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT64_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT64_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineParameterSize)} Elements	
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.25 Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01022] Definition of ImplementationDataType Dcm_StartFlexibleIn ArrayData_{Routine}_{Signal}Type

Name	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutine In/DcmDspStartRoutineInSignal.DcmDspRoutineParameterSize} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutine In/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStart RoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.26 Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01023] Definition of ImplementationDataType Dcm_StartFlexibleOut ArrayData_{Routine}_{Signal}Type

Name	Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{(ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutine Out/DcmDspStartRoutineOutSignal.DcmDspRoutineParameterSize} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutine Out/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStart RoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.27 Dcm_StopDataIn_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01024] Definition of ImplementationDataType Dcm_StopData In_{Routine}_{Signal}PrimitivType

Name	Dcm_StopDataIn_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	***Basetype***	***Variation***

▽

△

	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == FLOAT
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == DOUBLE
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT32
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT64
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT32
	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT64
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT8
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.28 Dcm_StopDataIn_{Routine}_{Signal}Type

[SWS_Dcm_91046] Definition of ImplementationDataType Dcm_StopDataIn_{Routine}_{Signal}Type

Name	Dcm_StopDataIn_{Routine}_{Signal}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***

▽

△

	Dcm_StopDataIn_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE))
	Dcm_StopDataIn_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N)
	Dcm_StopDataIn_{Routine}_{Signal}StructuredType	DcmDspStopRoutineInSignal with at least one DcmDspStopRoutineInSignalCompositeSignalRef.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.29 Dcm_StopDataIn_{Routine}_{Signal}ArrayType

[SWS_Dcm_91047] Definition of ImplementationDataType Dcm_StopDataIn_{Routine}_{Signal}ArrayType

Name	Dcm_StopDataIn_{Routine}_{Signal}ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == FLOAT_N
	float64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == DOUBLE_N)
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT64_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == UINT32_N)

▽

△

	uint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDsp RoutineSignalType)} == UINT64_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDsp RoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutine In/DcmDspStopRoutineInSignal.DcmDspRoutineParameterSize)} Elements	
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutine In/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.30 Dcm_StopDataOut_{Routine}_{Signal}PrimitivType

[SWS_Dcm_01025] Definition of ImplementationDataType Dcm_StopData Out_{Routine}_{Signal}PrimitivType

Name	Dcm_StopDataOut_{Routine}_{Signal}PrimitivType	
Kind	Type	
Derived from	***Basetype***	***Variation***
	float32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == FLOAT
	float64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == DOUBLE
	sint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == SINT16
	sint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == SINT32
	sint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == SINT64
	sint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == SINT8
	uint16	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == UINT16
	uint32	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp StopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.DcmDspRoutineSignalType)} == UINT32

▽

△

	uint64	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == UINT64
	uint8	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == UINT8
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.31 Dcm_StopDataOut_{Routine}_{Signal}Type

[SWS_Dcm_91048] Definition of ImplementationDataType Dcm_StopDataOut_{Routine}_{Signal}Type

Name	Dcm_StopDataOut_{Routine}_{Signal}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_StopDataOut_{Routine}_{Signal}PrimitivType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE))
	Dcm_StopDataOut_{Routine}_{Signal}ArrayType	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N)
	Dcm_StopDataOut_{Routine}_{Signal}StructuredType	DcmDspStopRoutineOutSignal with at least one DcmDspStopRoutineOutSignalCompositeSignalRef.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.32 Dcm_StopDataOut_{Routine}_{Signal}ArrayType

[SWS_Dcm_91049] Definition of ImplementationDataType Dcm_StopDataOut_{Routine}_{Signal}ArrayType

Name	Dcm_StopDataOut_{Routine}_{Signal}ArrayType	
Kind	Array	
Element type	***Type***	***Variation***
	float32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == FLOAT_N
	float64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == DOUBLE_N)
	sint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT16_N)
	sint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT32_N)
	sint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT64_N)
	sint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == SINT8_N)
	uint16	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == UINT16_N)
	uint32	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == UINT32_N)
	uint64	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == UINT64_N)
	uint8	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == UINT8_N)
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineParameterSize)} Elements	
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == ([U|S]INT[8|16|32|64]|FLOAT|DOUBLE)_N) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.1.33 Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01026] Definition of ImplementationDataType Dcm_StopFlexibleIn ArrayData_{Routine}_{Signal}Type

Name	Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutine In/DcmDspStopRoutineInSignal.DcmDspRoutineParameterSize} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutine In/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.34 Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type

[SWS_Dcm_01027] Definition of ImplementationDataType Dcm_StopFlexibleOut ArrayData_{Routine}_{Signal}Type

Name	Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutine Out/DcmDspStopRoutineOutSignal.DcmDspRoutineParameterSize} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutine Out/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRoutineUsePort)} == TRUE) Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.35 Dcm_KeyArray_{SecurityLevel}Type

[SWS_Dcm_01028] Definition of ImplementationDataType Dcm_KeyArray_{SecurityLevel}Type

Name	Dcm_KeyArray_{SecurityLevel}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityKey Size)} Elements		
Description	–		

▽

△

Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}
Available via	Rte_Dcm_Type.h

E.3.1.36 Dcm_SeedArray_{SecurityLevel}Type

[SWS_Dcm_01029] Definition of ImplementationDataType Dcm_SeedArray_{SecurityLevel}Type

Name	Dcm_SeedArray_{SecurityLevel}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecuritySeedSize)} Elements		
Description	–		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.37 Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type

[SWS_Dcm_01159] Definition of ImplementationDataType Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type

Name	Dcm_SecurityAccessDataRecordArray_{SecurityLevel}Type		
Kind	Array	***Element type***	uint8
Size	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp/DcmDspSecurity/DcmDspSecurityRow/DcmDspSecurityADRSize)} Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityUsePort)} == USE_ASYNCH_CLIENT_SERVER) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.DcmDspSecurityADRSize)} != NULL) SecurityLevel = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspSecurity/DcmDspSecurityRow.SHORT-NAME)}		
Available via	Rte_Dcm_Type.h		

E.3.1.38 Dcm_RequestDataArrayType

[SWS_Dcm_01165] Definition of ImplementationDataType Dcm_RequestDataArrayType

Name	Dcm_RequestDataArrayType		
Kind	Array	***Element type***	uint8
Size	(MAX({ecuc(Dcm/DcmConfigSet/DcmDsl/DcmDslProtocol/DcmDslProtocolRow/DcmDslProtocolRxBufferID->DcmDslBuffer.DcmDslBufferSize)}) -1) Elements		
Description	–		
Variation	–		
Available via	Rte_Dcm_Type.h		

E.3.1.39 Dcm_ControlMask_{DID}Type

[SWS_Dcm_01320] Definition of ImplementationDataType Dcm_ControlMask_{DID}Type

Name	Dcm_ControlMask_{DID}Type	
Kind	Type	
Derived from	***Basetype***	***Variation***
	Dcm_ControlMask_{Data}ArrayType	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} >= 0x05) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ SYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)))
	Dcm_ControlMask_{Data}_PrimitiveType	({ecuc(Dcm/DcmConfigSet/ DcmDsp/DcmDspDid/ DcmDspDidInfoRef->DcmDspDidInfo/ DcmDspDidControl/ DcmDspDidControlMaskSize)} <= 0x04)
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_SYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)} == DCM_CONTROLMASK_EXTERNAL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlEnableMask)} == NULL) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})	
Available via	Rte_Dcm_Type.h	

E.3.1.40 Dcm_InputOutputControlParameterType

[SWS_Dcm_01305] Definition of ImplementationDataType Dcm_InputOutputControlParameterType

Name	Dcm_InputOutputControlParameterType		
Kind	Type		
Derived from	uint8		
Range	DCM_RETURN_CONTROL_TO_ECU	0x00	returnControlToECU
	DCM_RESET_TO_DEFAULT	0x01	resetToDefault
	DCM_FREEZE_CURRENT_STATE	0x02	freezeCurrentState
	DCM_SHORT_TERM_ADJUSTMENT	0x03	shortTermAdjustment
	DCM_IDLE	0xff	Idle state, no request in processing (initial value)
Description	–		
Variation	–		
Available via	Rte_Dcm_Type.h		

E.3.1.41 Dcm_IOOperationRequest_{DID}Type

[SWS_Dcm_01306] Definition of ImplementationDataType Dcm_IOOperationRequest_{DID}Type

Name	Dcm_IOOperationRequest_{DID}Type	
Kind	Structure	
Elements	inputOutputControlParameter	
	Type	Dcm_InputOutputControlParameterType
	Comment	–
	controlEnableMask	
	Type	Dcm_Cemr_{DID}Type
	Comment	–
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE)) ||({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE))&& ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl)} != NULL) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})	
Available via	Rte_Dcm_Type.h	

E.3.1.42 Dcm_IOOperationResponseType

[SWS_Dcm_01307] Definition of ImplementationDataType Dcm_IOOperationResponseType

Name	Dcm_IOOperationResponseType		
Kind	Type		
Derived from	uint8		
Range	DCM_POSITIVE_RESPONSE	0x00	positive response (similar to E_OK)
	DCM_GENERAL_REJECT	0x10	NRC generalReject
	DCM_BUSY_REPEAT_REQUEST	0x21	NRC busyRepeatRequest
	DCM_CONDITIONS_NOT_CORRECT	0x22	NRC conditionsNotCorrect
	DCM_FAILURE_PREVENTS_EXECUTION	0x26	NRC FailurePreventsExecutionOfRequestedAction
	DCM_REQUEST_OUT_OF_RANGE	0x31	NRC requestOutOfRange
	DCM_RESPONSE_PENDING	0x78	ResponsePending (similar to E_PENDING)
Description	–		
Variation	–		
Available via	Rte_Dcm_Type.h		

E.3.1.43 Dcm_ResponseDataArrayType

[SWS_Dcm_91064] Definition of ImplementationDataType Dcm_ResponseDataArrayType

Name	Dcm_ResponseDataArrayType		
Kind	Array	***Element type***	uint8
Size	(MAX({ecuc(Dcm/DcmConfigSet/DcmDsl/DcmDslProtocol/DcmDslProtocolRow/DcmDslProtocolTxBufferRef->DcmDslBuffer.DcmDslBufferSize)}) -1) Elements		
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRequestFileTransfer/DcmRequestFileTransferUsePort)} == TRUE)		
Available via	Rte_Dcm_Type.h		

E.3.1.44 Dcm_ControlMask_{Data}ArrayType

[SWS_Dcm_91050] Definition of ImplementationDataType Dcm_ControlMask_{Data}ArrayType

Name	Dcm_ControlMask_{Data}ArrayType		
Kind	Array	***Element type***	uint8
Size	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)}) Elements		
Description	The current DcmDspData is referenced by the DcmDspDID.		
Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} >= 0x05) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.DcmDspDataUsePort)} == (USE_DATA_ SYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER || USE_DATA_ASYNCH_CLIENT_SERVER_ERROR))) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)})		
Available via	Rte_Dcm_Type.h		

E.3.1.45 Dcm_Cemr_{DID}Type

[SWS_Dcm_91087] Definition of ImplementationDataType Dcm_Cemr_{DID}Type

Name	Dcm_Cemr_{DID}Type	
Kind	Bitfield	
Derived from	Dcm_ControlMask_{DID}Type	
Elements	***Variable bit***	
	Kind	bit
	Name	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidControl. DcmDspDidControlEnableMask.SHORT-NAME)}
	Mask	2^ (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMaskSize)} * 8 - 1) - {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlEnableMask/DcmDspDidControlMaskBitPosition)})
	Description	Bitmask of data element in control enable mask record.
Description	–	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDidControl/DcmDspDidControlEnableMask)} != NULL) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}	
Available via	Rte_Dcm_Type.h	

E.3.2 Sender-Receiver-Interfaces

Using the concepts of the SWC template [17], the interface is defined as follows if SenderReceiver interface is used (DcmDspDataUsePort set to USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE):

E.3.2.1 DataServices_{DID}

[SWS_Dcm_91057] Definition of SenderReceiverInterface DataServices_{DID}

Name	DataServices_{DID}	
Comment	–	
IsService	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE)	
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE)) DID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}	
Data Elements	data	
	Type	{DID}_Struct_DataType
	Variation	DID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}

E.3.2.2 DataServices_{Data}

[SWS_Dcm_00687] Definition of SenderReceiverInterface DataServices_{Data}

Name	DataServices_{Data}	
Comment	–	
IsService	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} == USE_DATA_SENDER_RECEIVER_AS_SERVICE))	
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_DATA_ELEMENT_SPECIFIC_INTERFACES)&&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE))) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} ==(USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE)) Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME})	
Data Elements	data	
	Type	Dcm_DataElement_{Data}Type
	Variation	Data = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)})

E.3.2.3 IOControlRequest_{DID}

[SWS_Dcm_01308] Definition of SenderReceiverInterface IOControlRequest_{DID}

Name	IOControlRequest_{DID}	
Comment	Attention: controlState is only valid in case of IOOperationRequest is set to shortTerm Adjustment. The DCM provides a byte stream which could be transformed via transformer into an complex type.	
IsService	true	
Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE) ||({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE))&&({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/ DcmDspDidControl)}!= NULL)) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl/DcmDspDidControlMask)}== DCM_CONTROLMASK_EXTERNAL))) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})	
Data Elements	underControl	
	Type	Dcm_Cemr_{DID}Type
	Variation	–
	IOOperationRequest	
	Type	Dcm_IOOperationRequest_{DID}Type
	Variation	DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})
	controlState	
	Type	{DID}_Struct_DataType
	Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl/DcmDspDidShortTermAdjustment)} == True) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})

E.3.2.4 IOControlResponse_{DID}

[SWS_Dcm_01309] Definition of SenderReceiverInterface IOControlResponse_{DID}

Name	IOControlResponse_{DID}	
Comment	–	
IsService	true	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} ==(USE_ATOMIC_SENDER_RECEIVER_INTERFACE || USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE))) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl)} != NULL)) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})	
Data Elements	IOOperationResponse	
	Type	Dcm_IOOperationResponseType
	Variation	–

E.3.3 NvDataInterface

E.3.3.1 DataServices_{DID}

[SWS_Dcm_91061] Definition of SenderReceiverInterface DataServices_{DID}

Name	DataServices_{DID}	
Comment	–	
IsService	false	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_ATOMIC_NV_DATA_INTERFACE) DID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}	
Data Elements	data	
	Type	{DID}_Struct_DataType
	Variation	DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})

E.3.4 Client-Server-Interfaces

E.3.4.1 DataServices_{Data}

[SWS_Dem_00621] Definition of ClientServerInterface DataServices_{Data}

Name	DataServices_{Data}		
Comment	If configured it gets the data element value contained in a DID, a PID, or an extended data record from the respective SW-C via client/server or sender/receiver communication (refer to Figure "Dem and Dcm module requests PID data elements of SW-C via ReadData operation"). For each data element, one port of this interface type is provided by the SW-Cs.		
IsService	true		
Variation	(({ecuc(Dem/DemGeneral/DemDataElementClass)} instanceof {ecuc(Dem/DemGeneral/DemDataElementClass/DemExternalCSDataElementClass)}) && ({ecuc(Dem/DemGeneral/DemDataElementClass/DemExternalCSDataElementClass/DemDataElementUsePort)} == true)) Data = {ecuc(Dem/DemGeneral/DemDataElementClass.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	ReadData	
Comment	The server is not allowed to return E_NOT_OK, but shall always provide a valid data value (e.g. a default/replacement value in an error-case) to Dcm/Dem nevertheless the signature of the operation includes E_NOT_OK to ensure compatibility between server runnable and RTE Call API, since the RTE may return negative Std_Return values in certain cases (e.g. partition of server stopped)	
Relates to	<Module>_DemRead<DataElement>	
Variation	–	
Parameters	Data	
	Type	Dem_DataType_{Data}
	Direction	OUT
	Comment	–
	Variation	Data = {ecuc(Dem/DemGeneral/DemDataElementClass.SHORT-NAME)}
	monitorData0	

▽

△

	Type	Dem_MonitorDataType
	Direction	IN
	Comment	–
	Variation	({ecuc(Dem/DemGeneral/DemDataElementClass/DemExternalCSData ElementClass/DemDataElementProvideMonitorData)} == true))
Possible Errors	E_OK E_NOT_OK	

E.3.4.2 RoutineServices_{RoutineName}

The following interface defines operations needed for the UDS Service RoutineControl (0x31).

[SWS_Dcm_00690] Definition of ClientServerInterface RoutineServices_{Routine Name}

Name	RoutineServices_{RoutineName}		
Comment	–		
IsService	true		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineUsePort)} == TRUE RoutineName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	10	DCM_E_PENDING	Request is not yet finished. Further call(s) required to finish.
	12	DCM_E_FORCE_RCRRP	application request the transmission of a response Response Pending (NRC 0x78)

Operation	RequestResults	
Comment	–	
Relates to	Xxx_RequestResults	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/Dcm DspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutine SignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_RequestDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN

▽

△

	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_RequestDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequest RoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	RequestResults	
Comment	–	
Relates to	Xxx_RequestResults	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/Dcm DspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutine SignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_RequestDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_RequestDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–

▽

△

	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	DataOut_{Signal}	
	Type	Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	currentDataLength	
	Type	uint16
	Direction	OUT
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	–	

Operation	RequestResults	
Comment	–	
Relates to	Xxx_RequestResults	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_RequestDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequestRoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	DataIn_{Signal}	
	Type	Dcm_RequestFlexibleInArrayData_{Routine}_{Signal}Type
	Direction	IN
	Comment	–

▽

△

	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_RequestDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequest RoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	–	

Operation	RequestResults	
Comment	–	
Relates to	Xxx_RequestResults	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/Dcm DspRequestRoutineResultsOut/DcmDspRequestRoutineResultsOutSignal.DcmDspRoutine SignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine/DcmDspRequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_RequestDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsIn/DcmDspRequest RoutineResultsInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	DataIn_{Signal}	
	Type	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	–
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN

▽

△

	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_RequestDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutine Results/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResults OutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequest RoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	DataOut_{Signal}	
	Type	Dcm_RequestFlexibleOutArrayData_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutine Results/DcmDspRequestRoutineResultsOut/DcmDspRequestRoutineResults OutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDsp RequestRoutineResults/DcmDspRequestRoutineResultsOut/DcmDspRequest RoutineResultsOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	currentDataLength	
	Type	uint16
	Direction	OUT
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	RequestResultsConfirmation	
Comment	This operation indicates the transmission of a response to a RequestResultsRoutine request	
Relates to	Xxx_RequestResultsConfirmation	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspRequestRoutineResults/Dcm DspRequestRoutineResultsConfirmationEnabled)}==TRUE)	
Parameters	ConfirmationStatus	
	Type	Dcm_ConfirmationStatusType
	Direction	IN
	Comment	Confirmation status of a RequestResultsRoutinerequest
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK

Operation	Start	
Comment	–	
Relates to	Xxx_Start	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StartDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StartDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	Start
Comment	–
Relates to	Xxx_Start

▽

△

Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StartDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)} Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StartDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)
	DataOut_{Signal}	
	Type	Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	–
	currentDataLength	
	Type	uint16
	Direction	OUT
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	Start	
Comment	–	
Relates to	Xxx_Start	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StartDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	DataIn_{Signal}	
	Type	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	–
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StartDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	currentDataLength	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	Start	
Comment	–	
Relates to	Xxx_Start	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StartDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	DataIn_{Signal}	
	Type	Dcm_StartFlexibleInArrayData_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineIn/DcmDspStartRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StartDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	DataOut_{Signal}	
	Type	Dcm_StartFlexibleOutArrayData_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–

▽

△

	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineOut/DcmDspStartRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	currentDataLength	
	Type	uint16
	Direction	INOUT
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	StartConfirmation	
Comment	This operation indicates the transmission of a response to a StartRoutine request	
Relates to	Xxx_StartConfirmation	
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStartRoutine/DcmDspStartRoutineConfirmationEnabled)}==TRUE	
Parameters	ConfirmationStatus	
	Type	Dcm_ConfirmationStatusType
	Direction	IN
	Comment	Confirmation status of a StartRoutine request
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	Stop	
Comment	–	
Relates to	Xxx_Stop	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StopDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–

▽

△

	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StopDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	Stop	
Comment	–	
Relates to	Xxx_Stop	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_ LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StopDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	

▽

△

	Type	Dcm_StopDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	DataOut_{Signal}	
	Type	Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	–
	currentDataLength	
	Type	uint16
	Direction	OUT
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	Stop	
Comment	–	
Relates to	Xxx_Stop	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_ LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StopDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	DataIn_{Signal}	
	Type	Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	–

▽

△

	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StopDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineOut/DcmDspStopRoutineOut Signal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	currentDataLength	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	Stop	
Comment	–	
Relates to	Xxx_Stop	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStop RoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_ LENGTH)	
Parameters	DataIn_{Signal}	
	Type	Dcm_StopDataIn_{Routine}_{Signal}Type
	Direction	IN
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/ DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignal Type)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStop Routine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDsp Routine.SHORT-NAME)}
	DataIn_{Signal}	
	Type	Dcm_StopFlexibleInArrayData_{Routine}_{Signal}Type

▽

△

	Direction	IN
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineIn/DcmDspStopRoutineInSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	OpStatus	
	Type	Dcm_OpStatusType
	Direction	IN
	Comment	–
	Variation	–
	DataOut_{Signal}	
	Type	Dcm_StopDataOut_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} != VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	DataOut_{Signal}	
	Type	Dcm_StopFlexibleOutArrayData_{Routine}_{Signal}Type
	Direction	OUT
	Comment	–
	Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.DcmDspRoutineSignalType)} == VARIABLE_LENGTH Signal = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineOut/DcmDspStopRoutineOutSignal.SHORT-NAME)} Routine = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}
	currentDataLength	
	Type	uint16
	Direction	INOUT
	Comment	–
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DCM_E_PENDING DCM_E_FORCE_RCRRP	

Operation	StopConfirmation	
Comment	This operation indicates the transmission of a response to a StopRoutine request	
Relates to	Xxx_StopConfirmation	
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine/DcmDspStopRoutine/DcmDspStopRoutineConfirmationEnabled)}==TRUE)	
Parameters	ConfirmationStatus	
	Type	Dcm_ConfirmationStatusType
	Direction	IN
	Comment	Confirmation status of a StopRoutine request
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.3.4.3 ServiceRequestNotification

The following interface indicates to the Application that a service is about to be executed and allows the Application to reject the execution of the service request.

[SWS_Dcm_00694] Definition of ClientServerInterface ServiceRequestNotification

Name	ServiceRequestNotification		
Comment	–		
IsService	true		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestManufacturerNotification)} != NULL)||({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestSupplierNotification)} != NULL)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	8	E_REQUEST_NOT_ACCEPTED	no response will be sent

Operation	Confirmation	
Comment	–	
Relates to	Xxx_Confirmation	
Variation	–	
Parameters	SID	
	Type	uint8
	Direction	IN
	Comment	Value of service identifier
	Variation	–
	ReqType	
	Type	uint8
	Direction	IN
	Comment	Addressing type of the request(0=physical request, 1=functional request)
	Variation	–

▽

△

	ConnectionId	
	Type	uint16
	Direction	IN
	Comment	Unique connection identifier
	Variation	–
	ConfirmationStatus	
	Type	Dcm_ConfirmationStatusType
	Direction	IN
	Comment	Confirmation of a successful transmission or a transmission error of a diagnostic service.
	Variation	–
	ProtocolType	
	Type	Dcm_ProtocolType
	Direction	IN
	Comment	–
	Variation	–
	TesterSourceAddress	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	Indication	
Comment	–	
Relates to	Xxx_Indication	
Variation	–	
Parameters	SID	
	Type	uint8
	Direction	IN
	Comment	Value of service identifier
	Variation	–
	RequestData	
	Type	Dcm_RequestDataArrayType
	Direction	IN
	Comment	This parameter contains the complete request data (diagnostic buffer), except the service ID
	Variation	–
	DataSize	
	Type	uint32
	Direction	IN
	Comment	This parameter defines how many bytes in the RequestData parameter are valid
	Variation	–
	ReqType	
	Type	uint8
	Direction	IN

▽

△

	Comment	Addressing type of the request(0=physical request, 1=functional request
	Variation	–
	ConnectionId	
	Type	uint16
	Direction	IN
	Comment	Unique connection identifier
	Variation	–
	ErrorCode	
	Type	Dcm_NegativeResponseCodeType
	Direction	OUT
	Comment	–
	Variation	–
	ProtocolType	
	Type	Dcm_ProtocolType
	Direction	IN
	Comment	–
	Variation	–
	TesterSourceAddress	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK E_REQUEST_NOT_ACCEPTED	

E.3.5 Ports

E.3.5.1 DataServices_{DID} RPortProtoype

[SWS_Dcm_91059] Definition of Port DataServices_{DID} required by module Dcm

Name	DataServices_{DID}		
Kind	RequiredPort	***Interface***	DataServices_{DID}, DataServices_{DID}
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_NV_DATA_INTERFACE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidWrite)} == NULL) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidRead)} != NULL) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidControl)} !=NULL)) DID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}		

E.3.5.2 DataServices_{DID} PPortPrototype

[SWS_Dcm_91058] Definition of Port DataServices_{DID} provided by module Dcm

Name	DataServices_{DID}		
Kind	ProvidedPort	***Interface***	DataServices_{DID}, DataServices_{DID}
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_NV_DATA_INTERFACE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidWrite)} != NULL) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidRead)} == NULL) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl)} !=NULL)) DID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}		

E.3.5.3 DataServices_{DID} PRPortPrototype

[SWS_Dcm_91060] Definition of Port DataServices_{DID} provided/required by module Dcm

Name	DataServices_{DID}		
Kind	Provided RequiredPort	***Interface***	DataServices_{DID}, DataServices_{DID}
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE) ||({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE) || ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_NV_DATA_INTERFACE)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfo Ref->DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/Dcm DspDid/DcmDspDidInfoRef->DcmDspDidInfo/DcmDspDidRead)} != NULL)) DID = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)}		

E.3.5.4 DataServices_{Data} RPortPrototype

[SWS_Dcm_01035] Definition of Port DataServices_{Data} required by module Dcm

Name	DataServices_{Data}		
Kind	RequiredPort	***Interface***	DataServices_{Data}, DataServices_{Data}
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData/DcmDspPidService01/DcmDspPidDataUsePort)} ==(USE_DATA_SYNCH_CLIENT_SERVER ||USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE)) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)} || {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspPid/DcmDspPidData.SHORT-NAME)}		

E.3.5.5 DataServices_{Data} RPortProtoype

[SWS_Dcm_01310] Definition of Port DataServices_{Data} required by module Dcm

Name	DataServices_{Data}		
Kind	RequiredPort	***Interface***	DataServices_{Data}, DataServices_{Data}
Description	–		
Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_DATA_ELEMENT_SPECIFIC_INTERFACES)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE))) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidWrite)} == NULL) && (({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidRead)} != NULL))) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

E.3.5.6 DataServices_{Data} PPortProtoype

[SWS_Dcm_01031] Definition of Port DataServices_{Data} provided by module Dcm

Name	DataServices_{Data}		
Kind	ProvidedPort	***Interface***	DataServices_{Data}, DataServices_{Data}
Description	–		
Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_DATA_ELEMENT_SPECIFIC_INTERFACES)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE))) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidRead)} == NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

E.3.5.7 DataServices_{Data} PRPortProtoype

[SWS_Dcm_01311] Definition of Port DataServices_{Data} provided/required by module Dcm

Name	DataServices_{Data}		
Kind	Provided RequiredPort	***Interface***	DataServices_{Data}
Description	–		
Variation	((({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == USE_DATA_ELEMENT_SPECIFIC_INTERFACES)) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort)} == (USE_DATA_SENDER_RECEIVER ||USE_DATA_SENDER_RECEIVER_AS_SERVICE))) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidWrite)} != NULL) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidRead)} != NULL) Data = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspData.SHORT-NAME)}		

E.3.5.8 IOControlRequest_{DID}

[SWS_Dcm_01312] Definition of Port IOControlRequest_{DID} provided/required by module Dcm

Name	IOControlRequest_{DID}		
Kind	Provided RequiredPort	***Interface***	IOControlRequest_{DID}
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE || USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE))) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl)} != NULL) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})		

E.3.5.9 IOControlResponse_{DID}

[SWS_Dcm_01313] Definition of Port IOControlResponse_{DID} required by module Dcm

Name	IOControlResponse_{DID}		
Kind	RequiredPort	***Interface***	IOControlResponse_{DID}
Description	–		
Variation	(({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidUsePort)} == (USE_ATOMIC_SENDER_RECEIVER_INTERFACE || USE_ATOMIC_SENDER_RECEIVER_INTERFACE_AS_SERVICE))) && ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid/DcmDspDidInfoRef-> DcmDspDidInfo/DcmDspDidControl)} != NULL) DID = ({ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspDid.SHORT-NAME)})		

E.3.5.10 ServiceRequestManufacturerNotification_{Name}

[SWS_Dcm_01039] Definition of Port ServiceRequestManufacturerNotification_{Name} required by module Dcm

Name	ServiceRequestManufacturerNotification_{Name}		
Kind	RequiredPort	***Interface***	ServiceRequestNotification
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestManufacturerNotification)} != NULL) Name = {ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestManufacturerNotification.SHORT-NAME)}		

E.3.5.11 ServiceRequestSupplierNotification_{Name}

[SWS_Dcm_01042] Definition of Port ServiceRequestSupplierNotification_{Name} required by module Dcm

Name	ServiceRequestSupplierNotification_{Name}		
Kind	RequiredPort	***Interface***	ServiceRequestNotification
Description	–		
Variation	({ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestSupplierNotification)} != NULL) Name = {ecuc(Dcm/DcmConfigSet/DcmDsd/DcmDsdServiceRequestSupplier Notification.SHORT-NAME)}		

E.3.5.12 RoutineServices_{RoutineName}

[SWS_Dcm_01040] Definition of Port RoutineServices_{RoutineName} required by module Dcm

Name	RoutineServices_{RoutineName}		
Kind	RequiredPort	***Interface***	RoutineServices_{RoutineName}
Description	–		
Variation	{ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.DcmDspRoutineUsePort)} == TRUE RoutineName = {ecuc(Dcm/DcmConfigSet/DcmDsp/DcmDspRoutine.SHORT-NAME)}		

E.4 Dem

E.4.1 Implementation Data Types

E.4.1.1 Dem_DebouncingStateType

[SWS_Dem_01000] Definition of ImplementationDataType Dem_DebouncingStateType

Name	Dem_DebouncingStateType			
Kind	Bitfield			
Derived from	uint8			
Elements	***Kind***	***Name***	***Mask***	***Description***
	bit	DEM_TEMPORARILY_ DEFECTIVE	0x01	Bit 0: Temporarily Defective (corresponds to 0 < FDC < 127)
	bit	DEM_FINALLY_DEFECTIVE	0x02	Bit 1: finally Defective (corresponds to FDC = 127)
	bit	DEM_TEMPORARILY_HEALED	0x04	Bit 2: temporarily healed (corresponds to -128 < FDC < 0)

▽

△

	bit	DEM_TEST_COMPLETE	0x08	Bit 3: Test complete (corresponds to FDC = -128 or FDC = 127)
	bit	DEM_DTR_UPDATE	0x10	Bit 4: DTR Update (= Test complete && Debouncing complete && enable conditions / storage conditions fulfilled)
Description	–			
Variation	–			
Available via	Rte_Dem_Type.h			

E.4.1.2 Dem_DTCOriginType

[SWS_Dem_00934] Definition of ImplementationDataType Dem_DTCOriginType

Name	Dem_DTCOriginType		
Kind	Type		
Derived from	uint16		
Range	DEM_DTC_ORIGIN_ PRIMARY_MEMORY	0x0001	Event information located in the primary memory
	DEM_DTC_ORIGIN_ PERMANENT_MEMORY	0x0003	The Event information is located in the permanent memory
	DEM_DTC_ORIGIN_OBD_ RELEVANT_MEMORY	0x0004	OBD event information located in the primary memory.
	DEM_DTC_ORIGIN_ USERDEFINED_ MEMORY_<Name>	0x01XX	Event information located in the user defined memory, where XX is the configured DemUserDefined MemoryIdentifier in hexadecimal and <Name> is the Short-Name of the DemUserDefinedMemory.
Description	This enum is used to define the location of the events. The definition and use of the different memory types is OEM-specific.		
Variation	–		
Available via	Rte_Dem_Type.h		

E.4.1.3 Dem_EventIdType

[SWS_Dem_00925] Definition of ImplementationDataType Dem_EventIdType

Name	Dem_EventIdType		
Kind	Type		
Derived from	uint16		
Range	1..65535	–	Internal identifier of a diagnostic event Remark: 0 is not a valid value

▽

△

Description	Identification of an event by assigned EventId. The EventId is assigned by the Dem. Example: 1 refers to monitor x, 2 refers to monitor y, etc.
Variation	–
Available via	Rte_Dem_Type.h

E.4.1.4 Dem_EventStatusType

[SWS_Dem_00926] Definition of ImplementationDataType Dem_EventStatus Type

Name	Dem_EventStatusType		
Kind	Type		
Derived from	uint8		
Range	DEM_EVENT_STATUS_ PASSED	0x00	Monitor reports qualified test result passed.
	DEM_EVENT_STATUS_ FAILED	0x01	Monitor reports qualified test result failed.
	DEM_EVENT_STATUS_ PREPASSED	0x02	Monitor reports non-qualified test result pre-passed (debounced Dem-internally).
	DEM_EVENT_STATUS_ PREFAILED	0x03	Monitor reports non-qualified test result pre-failed (debounced Dem-internally).
	DEM_EVENT_STATUS_ FDC_THRESHOLD_ REACHED	0x04	Monitor triggers the storage of ExtendedDataRecords and FreezeFrames ON_FDC_ THRESHOLD.
		0x05 - 0xFF	reserved
Description	This type contains all monitor test result values, which can be reported via Dem_SetEventStatus().		
Variation	–		
Available via	Rte_Dem_Type.h		

E.4.1.5 Dem_DebounceResetStatusType

[SWS_Dem_00927] Definition of ImplementationDataType Dem_DebounceReset StatusType

Name	Dem_DebounceResetStatusType		
Kind	Type		
Derived from	uint8		
Range	DEM_DEBOUNCE_ STATUS_FREEZE	0x00	Freeze the internal debounce timer.
	DEM_DEBOUNCE_ STATUS_RESET	0x01	Reset the internal debounce counter/timer.
		0x02 - 0xFF	reserved

▽

△

Description	This type contains all definitions to control an internal debounce counter/timer via the function Dem_ResetEventDebounceStatus().
Variation	–
Available via	Rte_Dem_Type.h

E.4.1.6 Dem_DTCFormatType

[SWS_Dem_00933] Definition of ImplementationDataType Dem_DTCFormatType

Name	Dem_DTCFormatType		
Kind	Type		
Derived from	uint8		
Range	DEM_DTC_FORMAT_OBD	0	selects the 2-byte OBD DTC format (refer to configuration parameter DemObdDTC)
	DEM_DTC_FORMAT_UDS	1	selects the 3-byte UDS DTC format (refer to configuration parameter DemUdsDTC)
	DEM_DTC_FORMAT_ J1939	2	selects the merged SPN + FMI to 3-byte J1939 DTC format (refer to DemJ1939DTC)
	DEM_DTC_FORMAT_ OBD_3BYTE	3	Selects the 3-byte OBD DTC defined by DemDtcValue3Byte in case that SAE J2012 and UDS DTC format separation is used.
Description	This type is used to select the format of the DTC value.		
Variation	–		
Available via	Rte_Dem_Type.h		

E.4.1.7 Dem_InitMonitorReasonType

[SWS_Dem_00942] Definition of ImplementationDataType Dem_InitMonitorReasonType

Name	Dem_InitMonitorReasonType		
Kind	Type		
Derived from	uint8		
Range	DEM_INIT_MONITOR_ CLEAR	0x01	Event was cleared and all internal values and states are reset.
	DEM_INIT_MONITOR_ RESTART	0x02	Operation cycle of the event was restarted.
	DEM_INIT_MONITOR_ REENABLED	0x03	Enable conditions or DTC settings re-enabled.
	DEM_INIT_MONITOR_ STORAGE_REENABLED	0x04	Storage condition reenabled.
Description	(Re-)Initialization reason returned by the callback <Module>_DemInitMonitorFor<EventName>().		

▽

△

Variation	–
Available via	Rte_Dem_Type.h

E.4.1.8 Dem_MaxDataValueType

[SWS_Dem_01072] Definition of ImplementationDataType Dem_MaxDataValue Type

Name	Dem_MaxDataValueType		
Kind	Array	***Element type***	uint8
Size	size of largest Extended data class / Freeze frame record Elements		
Description	–		
Variation	–		
Available via	Rte_Dem_Type.h		

E.4.1.9 Dem_MonitorDataType

[SWS_Dem_91036] Definition of ImplementationDataType Dem_MonitorData Type

Name	Dem_MonitorDataType
Kind	Type
Derived from	uint32
Description	This type is used to pass monitoring data to the Dem.
Variation	–
Available via	Rte_Dem_Type.h

E.4.1.10 Dem_MonitorStatusType

[SWS_Dem_91005] Definition of ImplementationDataType Dem_MonitorStatus Type

Name	Dem_MonitorStatusType			
Kind	Bitfield			
Derived from	uint8			
Elements	***Kind***	***Name***	***Mask***	***Description***
	bit	DEM_MONITOR_STATUS_TF	0x01	Bit0: TestFailed
	bit	DEM_MONITOR_STATUS_ TNCTOC	0x02	Bit1: TestNotCompletedThis OperationCycle

▽

△

Description	This type contains possible monitor status values.
Variation	–
Available via	Rte_Dem_Type.h

E.4.1.11 Dem_UdsStatusByteType

[SWS_Dem_00928] Definition of ImplementationDataType Dem_UdsStatusByteType

Name	Dem_UdsStatusByteType			
Kind	Bitfield			
Derived from	uint8			
Lower limit	0x00			
Upper limit	0xFF			
Elements	***Kind***	***Name***	***Mask***	***Description***
	bit	DEM_UDS_STATUS_TF	0x01	bit 0: TestFailed
	bit	DEM_UDS_STATUS_TFTOC	0x02	bit 1: TestFailedThisOperation Cycle
	bit	DEM_UDS_STATUS_PDTC	0x04	bit 2: PendingDTC
	bit	DEM_UDS_STATUS_CDTC	0x08	bit 3: ConfirmedDTC
	bit	DEM_UDS_STATUS_TNCSLC	0x10	bit 4: TestNotCompletedSince LastClear
	bit	DEM_UDS_STATUS_TFSLC	0x20	bit 5: TestFailedSinceLastClear
	bit	DEM_UDS_STATUS_TNCTOC	0x40	bit 6: TestNotCompletedThis OperationCycle
	bit	DEM_UDS_STATUS_WIR	0x80	bit 7: WarningIndicator Requested
Description	In this data-type each bit has an individual meaning. The bit is set to 1 when the condition holds. For example, if the 2nd bit (0x02) is set to 1, this means that the test failed this operation cycle. If the bit is set to 0, it has not yet failed this cycle.			
Variation	–			
Available via	Rte_Dem_Type.h			

E.4.2 Client-Server-Interfaces

E.4.2.1 CallbackClearEventAllowed

[SWS_Dem_00620] Definition of ClientServerInterface CallbackClearEventAllowed

Name	CallbackClearEventAllowed
Comment	If configured, it gets the permission to clear a specific event from the SW-C. For each event, there can be one port of this interface type.

▽

△

IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	ClearEventAllowed	
Comment	–	
Relates to	<Module>_DemClearEventAllowed<ForCondition>	
Variation	–	
Parameters	Allowed	
	Type	boolean
	Direction	OUT
	Comment	True - clearance of event is allowed False - clearance of event is not allowed
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.2.2 CallbackEventDataChanged

[SWS_Dem_00618] Definition of ClientServerInterface CallbackEventDataChanged

Name	CallbackEventDataChanged		
Comment	If configured it triggers SW-Cs on event related data changes. For each event, there can be one port of this interface type.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful

Operation	EventDataChanged
Comment	–
Relates to	–
Variation	–
Possible Errors	–

E.4.2.3 CallbackEventUdsStatusChanged

[SWS_Dem_00615] Definition of ClientServerInterface CallbackEventUdsStatusChanged

Name	CallbackEventUdsStatusChanged
Comment	If configured it triggers SW-Cs on event status byte changes. For each event, there can be several ports of this interface type.

▽

△

IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful

Operation	CallbackEventUdsStatusChanged	
Comment	–	
Relates to	–	
Variation	–	
Parameters	EventStatusByteOld	
	Type	Dem_UdsStatusByteType
	Direction	IN
	Comment	–
	Variation	–
	EventStatusByteNew	
	Type	Dem_UdsStatusByteType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	–	

E.4.2.4 CallbackGetFaultDetectCounter

[SWS_Dem_00622] Definition of ClientServerInterface CallbackGetFaultDetect Counter

Name	CallbackGetFaultDetectCounter		
Comment	If configured it get the monitor-internal fault detection counter value of a specific event from the SW-C.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetFaultDetectionCounter	
Comment	–	
Relates to	<Module>_DemGetFaultDetectionCounter<ForEvent>	
Variation	–	
Parameters	FaultDetectionCounter	
	Type	sint8
	Direction	OUT
	Comment	Value of FaultDetectionCounter
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.2.5 CallbackInitMonitorForEvent

[SWS_Dem_00613] Definition of ClientServerInterface CallbackInitMonitorForEvent

Name	CallbackInitMonitorForEvent		
Comment	If configure it triggers an event-specific initialization of the monitor part of the SW-C). For each event, there can be one port of this interface type.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	InitMonitorForEvent	
Comment	–	
Relates to	<Module>_DemInitMonitorFor<EventName>	
Variation	–	
Parameters	InitMonitorReason	
	Type	Dem_InitMonitorReasonType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.2.6 CallbackMonitorStatusChange

[SWS_Dem_91011] Definition of ClientServerInterface CallbackMonitorStatusChange

Name	CallbackMonitorStatusChange		
Comment	If configured it triggers SW-Cs on monitor status changes. For each event, there can be several ports of this interface type.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful

Operation	MonitorStatusChanged
Comment	–
Relates to	–
Variation	–
Possible Errors	–

E.4.2.7 DiagnosticInfo

[SWS_Dem_00599] Definition of ClientServerInterface DiagnosticInfo

Name	DiagnosticInfo		
Comment	Provides the capability to obtain the event information. One port of this interface type is provided per diagnostic event by the Dem Service Component. It has EventId as a port-defined argument.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	10	DEM_E_NO_DTC_AVAILABLE	there is no DTC configured in the requested format
	14	DEM_E_NO_FDC_AVAILABLE	there is no fault detection counter available for the requested event
	21	DEM_BUFFER_TOO_SMALL	The provided buffer size is too small
	48	DEM_NO_SUCH_ELEMENT	The requested event data is not currently stored (but the request was valid) OR The requested record number is not supported by the event OR The requested DID is not supported by the freeze frame (GetEventFreezeFrameDataEx)

Operation	GetDTCOfEvent	
Comment	–	
Relates to	Dem_GetDTCOfEvent	
Variation	–	
Parameters	DTCFormat	
	Type	Dem_DTCFormatType
	Direction	IN
	Comment	–
	Variation	–
	DTCOfEvent	
	Type	uint32
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DEM_E_NO_DTC_AVAILABLE	

Operation	GetDebouncingOfEvent	
Comment	–	
Relates to	Dem_GetDebouncingOfEvent	
Variation	(({ecuc(Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass)} instanceof {ecuc(Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass/DemDebounceCounterBased)}) || ({ecuc(Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass)} instanceof {ecuc(Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass/DemDebounceTimeBased)}))	
Parameters	DebouncingState	
	Type	Dem_DebouncingStateType
	Direction	OUT

▽

△

	Comment	Bit 0 Temporarily Defective (corresponds to 0 < FDC < 127) Bit 1 finally Defective (corresponds to FDC = 127) Bit 2 temporarily healed (corresponds to -128 < FDC < 0) Bit 3 Test complete (corresponds to FDC = -128 or FDC = 127) Bit 4 DTR Update (= Test complete && Debouncing complete && enable conditions / storage conditions fulfilled)
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetEventAvailable	
Comment	–	
Relates to	Dem_GetEventAvailable	
Variation	–	
Parameters	AvailableStatus	
	Type	boolean*
	Direction	OUT
	Comment	TRUE if the event is available. FALSE if the event is not available.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetEventExtendedDataRecordEx	
Comment	–	
Relates to	Dem_GetEventExtendedDataRecordEx	
Variation	–	
Parameters	RecordNumber	
	Type	uint8
	Direction	IN
	Comment	–
	Variation	–
	DestBuffer	
	Type	Dem_MaxDataValueType
	Direction	OUT
	Comment	–
	Variation	–
	Bufsize	
	Type	uint16
	Direction	INOUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DEM_BUFFER_TOO_SMALL DEM_NO_SUCH_ELEMENT	

Operation	GetEventFreezeFrameDataEx
Comment	–
Relates to	Dem_GetEventFreezeFrameDataEx
Variation	–
Parameters	RecordNumber

▽

△

	Type	uint8
	Direction	IN
	Comment	–
	Variation	–
	DataId	
	Type	uint16
	Direction	IN
	Comment	–
	Variation	–
	DestBuffer	
	Type	Dem_MaxDataValueType
	Direction	OUT
	Comment	–
	Variation	–
	BufSize	
	Type	uint16
	Direction	INOUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DEM_BUFFER_TOO_SMALL DEM_NO_SUCH_ELEMENT	

Operation	GetEventUdsStatus	
Comment	Gets the current UDS status byte assigned to the DTC for the event	
Relates to	Dem_GetEventUdsStatus	
Variation	–	
Parameters	UDSStatusByte	
	Type	Dem_UdsStatusByteType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetFaultDetectionCounter	
Comment	–	
Relates to	Dem_GetFaultDetectionCounter	
Variation	–	
Parameters	FaultDetectionCounter	
	Type	sint8
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK DEM_E_NO_FDC_AVAILABLE	

Operation	GetMonitorStatus	
Comment	–	
Relates to	Dem_GetMonitorStatus	
Variation	–	
Parameters	MonitorStatus	
	Type	Dem_MonitorStatusType
	Direction	OUT
	Comment	Monitor status byte of the requested event. If the return value of the function call is E_NOT_OK, this parameter does not contain valid data.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.2.8 DiagnosticMonitor

[SWS_Dem_00598] Definition of ClientServerInterface DiagnosticMonitor

Name	DiagnosticMonitor		
Comment	Provide the capability to modify the event information. One port of this interface type is provided per application-related diagnostic event by the Dem Service Component. It has Event Id as a port-defined argument.		
IsService	true		
Variation	(({Dem/DemGeneral/DemStorageCondition.DemStorageConditionReplacementEvent Ref})!=({Dem/DemConfigSet/DemEventParameter}) && ({Dem/DemConfigSet/DemEvent Parameter/DemEventReportingType} == STANDARD_REPORTING))		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	ClearPrestoredFreezeFrame
Comment	–
Relates to	Dem_ClearPrestoredFreezeFrame
Variation	{ecuc(Dem/DemGeneral/DemMaxNumberPrestoredFF)} > 0
Possible Errors	E_OK E_NOT_OK

Operation	PrestoreFreezeFrame
Comment	–
Relates to	Dem_PrestoreFreezeFrame
Variation	{ecuc(Dem/DemGeneral/DemMaxNumberPrestoredFF)} > 0
Possible Errors	E_OK E_NOT_OK

Operation	ResetEventDebounceStatus
Comment	–
Relates to	Dem_ResetEventDebounceStatus
Variation	({ecuc(Dem/DemGeneral/DemDebounceCounterBasedSupport)} == true) || ({ecuc(Dem/Dem General/DemDebounceTimeBasedSupport)} == true)

▽

△

Parameters	DebounceResetStatus	
	Type	Dem_DebounceResetStatusType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	ResetEventStatus
Comment	–
Relates to	Dem_ResetEventStatus
Variation	–
Possible Errors	E_OK E_NOT_OK

Operation	ResetMonitorStatus
Comment	–
Relates to	Dem_ResetMonitorStatus
Variation	–
Possible Errors	E_OK E_NOT_OK

Operation	SetEventDisabled
Comment	–
Relates to	Dem_SetEventDisabled
Variation	{ecuc(Dem/DemGeneral.DemOBDSupport)} != DEM_OBD_NO_OBD_SUPPORT
Possible Errors	E_OK E_NOT_OK

Operation	SetEventStatus	
Comment	–	
Relates to	Dem_SetEventStatus	
Variation	–	
Parameters	EventStatus	
	Type	Dem_EventStatusType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.2.9 DiagnosticMonitor_MonitorData

[SWS_Dem_91038] Definition of ClientServerInterface DiagnosticMonitor_MonitorData

Name	DiagnosticMonitor_MonitorData		
Comment	Provide means to report diagnostic monitor status with monitor data.		
IsService	true		
Variation	({Dem/DemConfigSet/DemEventParameter/DemEventReportingType} == STANDARD_REPORTING_WITH_MONITOR_DATA)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetEventStatusWithMonitorData	
Comment	–	
Relates to	Dem_SetEventStatusWithMonitorData	
Variation	–	
Parameters	EventStatus	
	Type	Dem_EventStatusType
	Direction	IN
	Comment	–
	Variation	–
	monitorData0	
	Type	Dem_MonitorDataType
	Direction	IN
	Comment	–
	Variation	–
	monitorData1	
	Type	Dem_MonitorDataType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.2.10 EventStatus

[SWS_Dem_00838] Definition of ClientServerInterface EventStatus

Name	EventStatus		
Comment	Provides the capability modify the event status. One port of this interface type is provided per application-related diagnostic event by the Dem Service Component. It has EventId as a port-defined argument.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetWIRStatus
Comment	–
Relates to	Dem_SetWIRStatus

▽

△

Variation	–	
Parameters	WIRStatus	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.4.3 Ports

E.4.3.1 Event

[SWS_Dem_01037] Definition of Port Event_{Name} provided by module Dem

Name	Event_{Name}		
Kind	ProvidedPort	***Interface***	DiagnosticMonitor
Description	–		
Port Defined Argument Value(s)	***Type***	Dem_EventIdType	
	Value	{ecuc(Dem/DemConfigSet/DemEventParameter/DemEventId.value)}	
Variation	Name = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)}		

E.4.3.2 DiagnosticMonitor_MonitorData

[SWS_Dem_91039] Definition of Port DiagnosticMonitor_MonitorData_{Name} provided by module Dem

Name	DiagnosticMonitor_MonitorData_{Name}		
Kind	ProvidedPort	***Interface***	DiagnosticMonitor_MonitorData
Description	–		
Port Defined Argument Value(s)	***Type***	Dem_EventIdType	
	Value	{ecuc(Dem/DemConfigSet/DemEventParameter/DemEventId.value)}	
Variation	Name = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)}		

E.4.3.3 EventStatus

[SWS_Dem_01036] Definition of Port EventStatus_{Name} provided by module Dem

Name	EventStatus_{Name}		
Kind	ProvidedPort	***Interface***	EventStatus
Description	–		
Port Defined Argument Value(s)	***Type***	Dem_EventIdType	
	Value	{ecuc(Dem/DemConfigSet/DemEventParameter/DemEventId.value)}	
Variation	Name = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)}		

E.4.3.4 EventInfo

[SWS_Dem_01034] Definition of Port EventInfo_{Name} provided by module Dem

Name	EventInfo_{Name}		
Kind	ProvidedPort	***Interface***	DiagnosticInfo
Description	–		
Port Defined Argument Value(s)	***Type***	Dem_EventIdType	
	Value	{ecuc(Dem/DemConfigSet/DemEventParameter/DemEventId.value)}	
Variation	Name = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)}		

E.4.3.5 CBClrEvt

[SWS_Dem_91024] Definition of Port CBClrEvt_{Name} required by module Dem

Name	CBClrEvt_{Name}		
Kind	RequiredPort	***Interface***	CallbackClearEventAllowed
Description	–		
Variation	(({ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackClearEventAllowed)} != NULL) && ({ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackClearEventAllowed/DemCallback ClearEventAllowedFnc)} == NULL)) Name = {ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackClearEvent Allowed.SHORT-NAME)}		

E.4.3.6 CBDataEvt

[SWS_Dem_01003] Definition of Port CBDataEvt_{Name} required by module Dem

Name	CBDataEvt_{Name}		
Kind	RequiredPort	***Interface***	CallbackEventDataChanged
Description	–		
Variation	(({ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackEventDataChanged)} != NULL) && ({ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackEventDataChanged/DemCallbackEventDataChangedFnc)} == NULL)) Name = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)}		

E.4.3.7 CBFaultDetectCtr

[SWS_Dem_01004] Definition of Port CBFaultDetectCtr_{Name} required by module Dem

Name	CBFaultDetectCtr_{Name}		
Kind	RequiredPort	***Interface***	CallbackGetFaultDetectCounter
Description	–		
Variation	({ecuc(Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass/DemDebounceMonitorInternal/DemCallbackGetFDC/DemCallbackGetFDCFnc)} == NULL) Name = {ecuc(Dem/DemConfigSet/DemEventParameter/DemDebounceAlgorithmClass/DemDebounceMonitorInternal/DemCallbackGetFDC.SHORT-NAME)}		

E.4.3.8 CBInitEvt

[SWS_Dem_01005] Definition of Port CBInitEvt_{Name} required by module Dem

Name	CBInitEvt_{Name}		
Kind	RequiredPort	***Interface***	CallbackInitMonitorForEvent
Description	–		
Variation	(({ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackInitMForE)} != NULL) && ({ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackInitMForE/DemCallbackInitMForEFnc)} == NULL)) Name = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)}		

E.4.3.9 CBEventUdsStatusChanged

[SWS_Dem_01008] Definition of Port CBEventUdsStatusChanged_{EventName}_{CallbackName} required by module Dem

Name	CBEventUdsStatusChanged_{EventName}_{CallbackName}		
Kind	RequiredPort	***Interface***	CallbackEventUdsStatusChanged
Description	–		
Variation	EventName = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)} CallbackName = {ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackEventUdsStatus Changed.SHORT-NAME)}		

E.4.3.10 CBMonitorStatusChanged

[SWS_Dem_91012] Definition of Port CBMonitorStatusChanged_{EventName}_{CallbackName} required by module Dem

Name	CBMonitorStatusChanged_{EventName}_{CallbackName}		
Kind	RequiredPort	***Interface***	CallbackMonitorStatusChange
Description	–		
Variation	({ecuc(Dem/DemConfigSet/DemEventParameter.DemEventKind)} == DEM_EVENT_KIND_SWC) EventName = {ecuc(Dem/DemConfigSet/DemEventParameter.SHORT-NAME)} CallbackName = {ecuc(Dem/DemConfigSet/DemEventParameter/DemCallbackMonitorStatus Changed.SHORT-NAME)}		

E.5 FiM

E.5.1 Implementation Data Types

E.5.1.1 FiM_FunctionIdType

[SWS_Fim_00027] Definition of ImplementationDataType FiM_FunctionIdType

Name	FiM_FunctionIdType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	platform depended	
	uint8	platform depended	
Range	0..255, 0..65535	–	Identifier of functionality Configurable, size depends on System complexity. Remark: Not all numbers are valid. The FIM data generation tool shall only assign valid values.
Description	Type for the FunctionID		

▽

△

Variation	–
Available via	Rte_FiM_Type.h

E.5.2 Client-Server-Interfaces

E.5.2.1 FiM_FunctionInhibition

[SWS_Fim_00090] Definition of ClientServerInterface FunctionInhibition

Name	FunctionInhibition		
Comment	The service interface is used by the SW-C to query the permission state of a specific FID and its corresponding inhibiting events.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetFunctionPermission	
Comment	Get the permission state of the respective FID.	
Relates to	FiM_GetFunctionPermission	
Variation	–	
Parameters	Permission	
	Type	boolean
	Direction	OUT
	Comment	The permission of the requested FID. TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetInhibitingEvents	
Comment	Retrieves the list of Dem_EventIdTypes that are currently inhibiting the FID.	
Relates to	FiM_GetInhibitingEvents	
Variation	({ecuc(FiM/FiMGeneral/FiMInhibitionTrackingDepth)} > 0)	
Parameters	EventIdBuffer	
	Type	FiM_EventIdArrayType
	Direction	OUT
	Comment	Pointer to an array where the FiM will write the stored inhibiting Event IDs.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.5.2.2 FiM_ControlFunctionAvailable

[SWS_Fim_00107] Definition of ClientServerInterface ControlFunctionAvailable

Name	ControlFunctionAvailable		
Comment	SW Components can use this service to set the availability of a function.		
IsService	true		
Variation	({ecuc(FiM/FiMGeneral/FiMAvailabilitySupport)} == True)		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetFunctionAvailable	
Comment	Sets the availability of a function.	
Relates to	FiM_SetFunctionAvailable	
Variation	–	
Parameters	Availability	
	Type	boolean
	Direction	IN
	Comment	The permission of the requested FID: TRUE: Function is available. FALSE: Function is not available.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

E.5.3 Ports

E.5.3.1 FiM_FunctionInhibition

[SWS_Fim_00094] Definition of Port Func_{Name} provided by module FiM

Name	Func_{Name}		
Kind	ProvidedPort	***Interface***	FunctionInhibition
Description	A client can query the FiM for execution permission for a specific function. The FIDs which represent the functions are not directly used by the client SW-C. Instead, the mechanism of "port-defined argument values" is used and every FID is mapped to a separate port that is responsible for the data exchange via RTE.		
Port Defined Argument Value(s)	***Type***	FiM_FunctionIdType	
	Value	{ecuc(FiM/FiMConfigSet/FiMFID/FiMFunctionId.value)}	
Variation	Name = {ecuc(FiM/FiMConfigSet/FiMFID.SHORT-NAME)}		

E.5.3.2 FiM_ControlFunctionAvailable

[SWS_Fim_00108] Definition of Port Control_{Name} provided by module FiM

Name	Control_{Name}		
Kind	ProvidedPort	***Interface***	ControlFunctionAvailable
Description	A client can set the availability for a specific function.		
Port Defined Argument Value(s)	***Type***	FiM_FunctionIdType	
	Value	{ecuc(FiM/FiMConfigSet/FiMFID/FiMFunctionId.value)}	
Variation	({ecuc(FiM/FiMGeneral/FiMAvailabilitySupport)} == True) Name = {ecuc(FiM/FiMConfigSet/FiMFID.SHORT-NAME)}		

F Changes History

F.1 Changes in R20-11 compared to R19-11

Please note that the document was intialy created in R20-11.

F.1.1 Added Specification Items in R20-11

[SWS_SwCluC_00001] [SWS_SwCluC_00002] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00013] [SWS_SwCluC_00014] [SWS_SwCluC_00015] [SWS_SwCluC_00016] [SWS_SwCluC_00017] [SWS_SwCluC_00018] [SWS_SwCluC_00019] [SWS_SwCluC_00020] [SWS_SwCluC_00021] [SWS_SwCluC_00022] [SWS_SwCluC_00023] [SWS_SwCluC_00024] [SWS_SwCluC_00025] [SWS_SwCluC_00026] [SWS_SwCluC_00027] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00036] [SWS_SwCluC_00037] [SWS_SwCluC_00038] [SWS_SwCluC_00039] [SWS_SwCluC_00040] [SWS_SwCluC_00041] [SWS_SwCluC_00042] [SWS_SwCluC_00043] [SWS_SwCluC_00044] [SWS_SwCluC_00045] [SWS_SwCluC_00046] [SWS_SwCluC_00047] [SWS_SwCluC_00048] [SWS_SwCluC_00049] [SWS_SwCluC_00050] [SWS_SwCluC_00051] [SWS_SwCluC_00052] [SWS_SwCluC_00053] [SWS_SwCluC_00054] [SWS_SwCluC_00055] [SWS_SwCluC_00056] [SWS_SwCluC_00057] [SWS_SwCluC_00058] [SWS_SwCluC_00059] [SWS_SwCluC_00060] [SWS_SwCluC_00061] [SWS_SwCluC_00062] [SWS_SwCluC_00063] [SWS_SwCluC_00064] [SWS_SwCluC_00065] [SWS_SwCluC_00066] [SWS_SwCluC_00067] [SWS_SwCluC_00068] [SWS_SwCluC_00069] [SWS_SwCluC_00070] [SWS_SwCluC_00071] [SWS_SwCluC_00072] [SWS_SwCluC_00075] [SWS_SwCluC_00076] [SWS_SwCluC_00077] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00084] [SWS_SwCluC_00085] [SWS_SwCluC_00086] [SWS_SwCluC_00088] [SWS_SwCluC_00089] [SWS_SwCluC_00999] [SWS_SwCluC_01000] [SWS_SwCluC_01002] [SWS_SwCluC_02000] [SWS_SwCluC_02001] [SWS_SwCluC_02002] [SWS_SwCluC_02003] [SWS_SwCluC_02004] [SWS_SwCluC_02005] [SWS_SwCluC_02006] [SWS_SwCluC_02007] [SWS_SwCluC_02008] [SWS_SwCluC_02101] [SWS_SwCluC_02102] [SWS_SwCluC_02103] [SWS_SwCluC_02104] [SWS_SwCluC_02105] [SWS_SwCluC_02106] [SWS_SwCluC_02107] [SWS_SwCluC_02108] [SWS_SwCluC_02109] [SWS_SwCluC_02110] [SWS_SwCluC_02111] [SWS_SwCluC_02112] [SWS_SwCluC_02113] [SWS_SwCluC_02114] [SWS_SwCluC_02115] [SWS_SwCluC_02116] [SWS_SwCluC_02117] [SWS_SwCluC_02118] [SWS_SwCluC_02119] [SWS_SwCluC_02120] [SWS_SwCluC_02121] [SWS_SwCluC_02122] [SWS_SwCluC_02123] [SWS_SwCluC_02124] [SWS_SwCluC_02125] [SWS_SwCluC_02126] [SWS_SwCluC_02127] [SWS_SwCluC_02128] [SWS_SwCluC_02129] [SWS_SwCluC_02132] [SWS_SwCluC_02133] [SWS_SwCluC_02136]

[SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02140] [SWS_SwCluC_02142] [SWS_SwCluC_02143] [SWS_SwCluC_02144] [SWS_SwCluC_02200] [SWS_SwCluC_02201] [SWS_SwCluC_02202] [SWS_SwCluC_02204] [SWS_SwCluC_02205] [SWS_SwCluC_02206] [SWS_SwCluC_02210] [SWS_SwCluC_02211] [SWS_SwCluC_02212] [SWS_SwCluC_02213] [SWS_SwCluC_02214] [SWS_SwCluC_02215] [SWS_SwCluC_02216] [SWS_SwCluC_02217] [SWS_SwCluC_02218] [SWS_SwCluC_02219] [SWS_SwCluC_02220] [SWS_SwCluC_02221] [SWS_SwCluC_02222] [SWS_SwCluC_02223] [SWS_SwCluC_02224] [SWS_SwCluC_02225] [SWS_SwCluC_02226] [SWS_SwCluC_02227] [SWS_SwCluC_02229] [SWS_SwCluC_02230] [SWS_SwCluC_02250] [SWS_SwCluC_02251] [SWS_SwCluC_02252] [SWS_SwCluC_02253] [SWS_SwCluC_02254] [SWS_SwCluC_02255] [SWS_SwCluC_02256] [SWS_SwCluC_02257] [SWS_SwCluC_02258] [SWS_SwCluC_02259] [SWS_SwCluC_02260] [SWS_SwCluC_02261] [SWS_SwCluC_02262] [SWS_SwCluC_02263] [SWS_SwCluC_02264] [SWS_SwCluC_02265] [SWS_SwCluC_02266] [SWS_SwCluC_03000] [SWS_SwCluC_03001] [SWS_SwCluC_03002] [SWS_SwCluC_03003] [SWS_SwCluC_03004] [SWS_SwCluC_03005] [SWS_SwCluC_03006] [SWS_SwCluC_03007] [SWS_SwCluC_03008] [SWS_SwCluC_03009] [SWS_SwCluC_03010] [SWS_SwCluC_03011] [SWS_SwCluC_03012] [SWS_SwCluC_03013] [SWS_SwCluC_03015] [SWS_SwCluC_03016] [SWS_SwCluC_03017] [SWS_SwCluC_03018] [SWS_SwCluC_03019] [SWS_SwCluC_03022] [SWS_SwCluC_03023] [SWS_SwCluC_03024] [SWS_SwCluC_03025] [SWS_SwCluC_03026] [SWS_SwCluC_03027] [SWS_SwCluC_03028] [SWS_SwCluC_03029] [SWS_SwCluC_03030] [SWS_SwCluC_03031] [SWS_SwCluC_03033] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03045] [SWS_SwCluC_03046] [SWS_SwCluC_03047] [SWS_SwCluC_03048] [SWS_SwCluC_03049] [SWS_SwCluC_03050] [SWS_SwCluC_03051] [SWS_SwCluC_03052] [SWS_SwCluC_03055] [SWS_SwCluC_03056] [SWS_SwCluC_03057] [SWS_SwCluC_03058] [SWS_SwCluC_03060] [SWS_SwCluC_03061] [SWS_SwCluC_03062] [SWS_SwCluC_03063] [SWS_SwCluC_03064] [SWS_SwCluC_03065] [SWS_SwCluC_03068] [SWS_SwCluC_03071] [SWS_SwCluC_03072] [SWS_SwCluC_03073] [SWS_SwCluC_03074] [SWS_SwCluC_03075] [SWS_SwCluC_03076] [SWS_SwCluC_03077] [SWS_SwCluC_03101] [SWS_SwCluC_03102] [SWS_SwCluC_03103] [SWS_SwCluC_03106] [SWS_SwCluC_03107] [SWS_SwCluC_03108] [SWS_SwCluC_03109] [SWS_SwCluC_03110] [SWS_SwCluC_03111] [SWS_SwCluC_03112] [SWS_SwCluC_03113] [SWS_SwCluC_03120] [SWS_SwCluC_03121] [SWS_SwCluC_03122] [SWS_SwCluC_03123] [SWS_SwCluC_03124] [SWS_SwCluC_03126] [SWS_SwCluC_03130] [SWS_SwCluC_03131] [SWS_SwCluC_03132] [SWS_SwCluC_03133] [SWS_SwCluC_03134] [SWS_SwCluC_03135] [SWS_SwCluC_03136] [SWS_SwCluC_03137] [SWS_SwCluC_03138] [SWS_SwCluC_03139] [SWS_SwCluC_03140] [SWS_SwCluC_03142] [SWS_SwCluC_03143] [SWS_SwCluC_03144] [SWS_SwCluC_03145] [SWS_SwCluC_03146] [SWS_SwCluC_03147] [SWS_SwCluC_03150] [SWS_SwCluC_03151] [SWS_SwCluC_03152] [SWS_SwCluC_03153] [SWS_SwCluC_03154] [SWS_SwCluC_03155] [SWS_SwCluC_03156] [SWS_SwCluC_03157] [SWS_SwCluC_03158] [SWS_SwCluC_03159]

[SWS_SwCluC_03160] [SWS_SwCluC_03161] [SWS_SwCluC_03162] [SWS_SwCluC_03163] [SWS_SwCluC_03164] [SWS_SwCluC_03165] [SWS_SwCluC_03166] [SWS_SwCluC_03167] [SWS_SwCluC_03168] [SWS_SwCluC_03169] [SWS_SwCluC_03170] [SWS_SwCluC_03171] [SWS_SwCluC_03173] [SWS_SwCluC_03174] [SWS_SwCluC_03175] [SWS_SwCluC_03176] [SWS_SwCluC_03177] [SWS_SwCluC_10000] [SWS_SwCluC_10001] [SWS_SwCluC_10002] [SWS_SwCluC_10003] [SWS_SwCluC_10004] [SWS_SwCluC_10005] [SWS_SwCluC_10006] [SWS_SwCluC_10007] [SWS_SwCluC_10008] [SWS_SwCluC_10009] [SWS_SwCluC_10010] [SWS_SwCluC_10011] [SWS_SwCluC_10020] [SWS_SwCluC_10021] [SWS_SwCluC_10022] [SWS_SwCluC_10023] [SWS_SwCluC_10032] [SWS_SwCluC_10033] [SWS_SwCluC_10034] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100] [SWS_SwCluC_90000] [SWS_SwCluC_90001] [SWS_SwCluC_90002] [SWS_SwCluC_90003] [SWS_SwCluC_90004] [SWS_SwCluC_90005] [SWS_SwCluC_90006] [SWS_SwCluC_90007] [SWS_SwCluC_90008] [SWS_SwCluC_90009] [SWS_SwCluC_90010] [SWS_SwCluC_CONSTR_00073] [SWS_SwCluC_CONSTR_00074] [SWS_SwCluC_CONSTR_00078] [SWS_SwCluC_CONSTR_00087] [SWS_SwCluC_CONSTR_02130] [SWS_SwCluC_CONSTR_02131] [SWS_SwCluC_CONSTR_02134] [SWS_SwCluC_CONSTR_02135] [SWS_SwCluC_CONSTR_02141] [SWS_SwCluC_CONSTR_02203] [SWS_SwCluC_CONSTR_02228] [SWS_SwCluC_CONSTR_02282] [SWS_SwCluC_CONSTR_02283] [SWS_SwCluC_CONSTR_02231] [SWS_SwCluC_CONSTR_02232] [SWS_SwCluC_CONSTR_02233] [SWS_SwCluC_CONSTR_02234] [SWS_SwCluC_CONSTR_02235] [SWS_SwCluC_CONSTR_02236] [SWS_SwCluC_CONSTR_02237] [SWS_SwCluC_CONSTR_03020] [SWS_SwCluC_CONSTR_03021] [SWS_SwCluC_CONSTR_03032] [SWS_SwCluC_CONSTR_03053] [SWS_SwCluC_CONSTR_03054] [SWS_SwCluC_CONSTR_03059] [SWS_SwCluC_CONSTR_03063] [SWS_SwCluC_CONSTR_03066] [SWS_SwCluC_CONSTR_03067] [SWS_SwCluC_CONSTR_03069]

F.1.2 Changed Specification Items in R20-11

none

F.1.3 Deleted Specification Items in R20-11

none

F.2 Changes in R21-11 compared to R20-11

F.2.1 Added Specification Items in R21-11

[SWS_SwCluC_00090] [SWS_SwCluC_02270] [SWS_SwCluC_02271] [SWS_SwCluC_02272] [SWS_SwCluC_02273] [SWS_SwCluC_02274] [SWS_SwCluC_02275] [SWS_SwCluC_02276] [SWS_SwCluC_02277] [SWS_SwCluC_02300] [SWS_SwCluC_02301] [SWS_SwCluC_02302] [SWS_SwCluC_02303] [SWS_SwCluC_02304] [SWS_SwCluC_02305] [SWS_SwCluC_02306] [SWS_SwCluC_02307] [SWS_SwCluC_02308] [SWS_SwCluC_02309] [SWS_SwCluC_02310] [SWS_SwCluC_02311] [SWS_SwCluC_02312] [SWS_SwCluC_02313] [SWS_SwCluC_02314] [SWS_SwCluC_02315] [SWS_SwCluC_02316] [SWS_SwCluC_02317] [SWS_SwCluC_02318] [SWS_SwCluC_02319] [SWS_SwCluC_02320] [SWS_SwCluC_02321] [SWS_SwCluC_02322] [SWS_SwCluC_02323] [SWS_SwCluC_02324] [SWS_SwCluC_02325] [SWS_SwCluC_02330] [SWS_SwCluC_02331] [SWS_SwCluC_02332] [SWS_SwCluC_02333] [SWS_SwCluC_02334] [SWS_SwCluC_02335] [SWS_SwCluC_02336] [SWS_SwCluC_02337] [SWS_SwCluC_02338] [SWS_SwCluC_02339] [SWS_SwCluC_02341] [SWS_SwCluC_02342] [SWS_SwCluC_02350] [SWS_SwCluC_02351] [SWS_SwCluC_02352] [SWS_SwCluC_02353] [SWS_SwCluC_02354] [SWS_SwCluC_02355] [SWS_SwCluC_02356] [SWS_SwCluC_02357] [SWS_SwCluC_02358] [SWS_SwCluC_02359] [SWS_SwCluC_02360] [SWS_SwCluC_02361] [SWS_SwCluC_02362] [SWS_SwCluC_02363] [SWS_SwCluC_02364] [SWS_SwCluC_02365] [SWS_SwCluC_02366] [SWS_SwCluC_02367] [SWS_SwCluC_02368] [SWS_SwCluC_02369] [SWS_SwCluC_02370] [SWS_SwCluC_02371] [SWS_SwCluC_02372] [SWS_SwCluC_02373] [SWS_SwCluC_02374] [SWS_SwCluC_02375] [SWS_SwCluC_02376] [SWS_SwCluC_02380] [SWS_SwCluC_02381] [SWS_SwCluC_02382] [SWS_SwCluC_02383] [SWS_SwCluC_02384] [SWS_SwCluC_02385] [SWS_SwCluC_02386] [SWS_SwCluC_02387] [SWS_SwCluC_02388] [SWS_SwCluC_02390] [SWS_SwCluC_02391] [SWS_SwCluC_02392] [SWS_SwCluC_02393] [SWS_SwCluC_02394] [SWS_SwCluC_02395] [SWS_SwCluC_02396] [SWS_SwCluC_02397] [SWS_SwCluC_02400] [SWS_SwCluC_02401] [SWS_SwCluC_02402] [SWS_SwCluC_02403] [SWS_SwCluC_02404] [SWS_SwCluC_02405] [SWS_SwCluC_02406] [SWS_SwCluC_02407] [SWS_SwCluC_02408] [SWS_SwCluC_02409] [SWS_SwCluC_02410] [SWS_SwCluC_02411] [SWS_SwCluC_02412] [SWS_SwCluC_02413] [SWS_SwCluC_02414] [SWS_SwCluC_02415] [SWS_SwCluC_02416] [SWS_SwCluC_02417] [SWS_SwCluC_02418] [SWS_SwCluC_02419] [SWS_SwCluC_02420] [SWS_SwCluC_02421] [SWS_SwCluC_02422] [SWS_SwCluC_02423] [SWS_SwCluC_02424] [SWS_SwCluC_02425] [SWS_SwCluC_02426] [SWS_SwCluC_02500] [SWS_SwCluC_02501] [SWS_SwCluC_02502] [SWS_SwCluC_02503] [SWS_SwCluC_02504] [SWS_SwCluC_02505] [SWS_SwCluC_02506] [SWS_SwCluC_02507] [SWS_SwCluC_02508] [SWS_SwCluC_02509] [SWS_SwCluC_02510] [SWS_SwCluC_02511] [SWS_SwCluC_02512] [SWS_SwCluC_02513] [SWS_SwCluC_02514] [SWS_SwCluC_02515] [SWS_SwCluC_02516] [SWS_SwCluC_02517] [SWS_SwCluC_02518] [SWS_SwCluC_02519] [SWS_SwCluC_02520] [SWS_SwCluC_02521] [SWS_SwCluC_02522] [SWS_SwCluC_02523] [SWS_SwCluC_02524]

[SWS_SwCluC_02525] [SWS_SwCluC_02526] [SWS_SwCluC_02527] [SWS_SwCluC_02528] [SWS_SwCluC_02529] [SWS_SwCluC_02530] [SWS_SwCluC_02531] [SWS_SwCluC_02532] [SWS_SwCluC_02533] [SWS_SwCluC_02534] [SWS_SwCluC_02535] [SWS_SwCluC_02536] [SWS_SwCluC_02537] [SWS_SwCluC_02538] [SWS_SwCluC_02539] [SWS_SwCluC_02540] [SWS_SwCluC_02541] [SWS_SwCluC_02542] [SWS_SwCluC_02543] [SWS_SwCluC_02544] [SWS_SwCluC_02545] [SWS_SwCluC_02548] [SWS_SwCluC_02549] [SWS_SwCluC_02550] [SWS_SwCluC_02551] [SWS_SwCluC_02552] [SWS_SwCluC_02600] [SWS_SwCluC_02601] [SWS_SwCluC_02602] [SWS_SwCluC_02603] [SWS_SwCluC_02604] [SWS_SwCluC_02605] [SWS_SwCluC_02606] [SWS_SwCluC_02607] [SWS_SwCluC_02608] [SWS_SwCluC_02609] [SWS_SwCluC_02610] [SWS_SwCluC_02611] [SWS_SwCluC_02612] [SWS_SwCluC_02613] [SWS_SwCluC_02614] [SWS_SwCluC_02615] [SWS_SwCluC_02616] [SWS_SwCluC_02617] [SWS_SwCluC_02618] [SWS_SwCluC_02619] [SWS_SwCluC_02620] [SWS_SwCluC_02621] [SWS_SwCluC_02622] [SWS_SwCluC_02623] [SWS_SwCluC_02624] [SWS_SwCluC_02625] [SWS_SwCluC_02626] [SWS_SwCluC_02627] [SWS_SwCluC_02628] [SWS_SwCluC_02629] [SWS_SwCluC_02630] [SWS_SwCluC_02631] [SWS_SwCluC_02632] [SWS_SwCluC_02633] [SWS_SwCluC_02634] [SWS_SwCluC_02636] [SWS_SwCluC_02637] [SWS_SwCluC_02638] [SWS_SwCluC_02639] [SWS_SwCluC_02640] [SWS_SwCluC_02642] [SWS_SwCluC_02643] [SWS_SwCluC_02644] [SWS_SwCluC_02645] [SWS_SwCluC_02646] [SWS_SwCluC_02647] [SWS_SwCluC_02648] [SWS_SwCluC_02649] [SWS_SwCluC_02650] [SWS_SwCluC_02651] [SWS_SwCluC_02652] [SWS_SwCluC_02661] [SWS_SwCluC_02662] [SWS_SwCluC_02663] [SWS_SwCluC_02700] [SWS_SwCluC_02701] [SWS_SwCluC_02702] [SWS_SwCluC_02703] [SWS_SwCluC_02704] [SWS_SwCluC_02705] [SWS_SwCluC_02706] [SWS_SwCluC_02707] [SWS_SwCluC_02708] [SWS_SwCluC_02709] [SWS_SwCluC_02710] [SWS_SwCluC_02711] [SWS_SwCluC_02712] [SWS_SwCluC_02713] [SWS_SwCluC_02714] [SWS_SwCluC_02715] [SWS_SwCluC_02716] [SWS_SwCluC_02717] [SWS_SwCluC_02718] [SWS_SwCluC_02719] [SWS_SwCluC_02723] [SWS_SwCluC_02724] [SWS_SwCluC_02725] [SWS_SwCluC_02726] [SWS_SwCluC_02727] [SWS_SwCluC_02728] [SWS_SwCluC_02729] [SWS_SwCluC_02730] [SWS_SwCluC_02731] [SWS_SwCluC_02732] [SWS_SwCluC_02733] [SWS_SwCluC_02734] [SWS_SwCluC_02735] [SWS_SwCluC_02736] [SWS_SwCluC_02737] [SWS_SwCluC_02738] [SWS_SwCluC_02739] [SWS_SwCluC_02740] [SWS_SwCluC_02741] [SWS_SwCluC_02742] [SWS_SwCluC_02743] [SWS_SwCluC_02744] [SWS_SwCluC_02745] [SWS_SwCluC_02746] [SWS_SwCluC_02747] [SWS_SwCluC_02748] [SWS_SwCluC_02749] [SWS_SwCluC_02750] [SWS_SwCluC_02751] [SWS_SwCluC_03080] [SWS_SwCluC_03081] [SWS_SwCluC_03082] [SWS_SwCluC_03200] [SWS_SwCluC_03201] [SWS_SwCluC_03202] [SWS_SwCluC_03203] [SWS_SwCluC_03204] [SWS_SwCluC_03205] [SWS_SwCluC_03206] [SWS_SwCluC_03207] [SWS_SwCluC_03208] [SWS_SwCluC_03209] [SWS_SwCluC_03210] [SWS_SwCluC_03211] [SWS_SwCluC_03212] [SWS_SwCluC_03213] [SWS_SwCluC_03214] [SWS_SwCluC_03215] [SWS_SwCluC_03216] [SWS_SwCluC_03217] [SWS_SwCluC_03218] [SWS_SwCluC_12200] [SWS_SwCluC_12300] [SWS_SwCluC_12400] [SWS_SwCluC_13000] [SWS_SwCluC_13001] [SWS_SwCluC_13002] [SWS_SwCluC_13003] [SWS_SwCluC_13004] [SWS_SwCluC_13005] [SWS_SwCluC_13006]

[SWS_SwCluC_13007] [SWS_SwCluC_13008] [SWS_SwCluC_13009] [SWS_SwCluC_13010] [SWS_SwCluC_13011] [SWS_SwCluC_13012] [SWS_SwCluC_13013] [SWS_SwCluC_13014] [SWS_SwCluC_13015] [SWS_SwCluC_90011] [SWS_SwCluC_90012] [SWS_SwCluC_90013] [SWS_SwCluC_90014] [SWS_SwCluC_90015] [SWS_SwCluC_90016] [SWS_SwCluC_90017] [SWS_SwCluC_90018] [SWS_SwCluC_90019] [SWS_SwCluC_90020] [SWS_SwCluC_90021] [SWS_SwCluC_90022] [SWS_SwCluC_CONSTR_00091] [SWS_SwCluC_CONSTR_00092] [SWS_SwCluC_CONSTR_00093] [SWS_SwCluC_CONSTR_00094] [SWS_SwCluC_CONSTR_02427] [SWS_SwCluC_CONSTR_02428] [SWS_SwCluC_CONSTR_02429] [SWS_SwCluC_CONSTR_02546] [SWS_SwCluC_CONSTR_02547] [SWS_SwCluC_CONSTR_02553] [SWS_SwCluC_CONSTR_02554] [SWS_SwCluC_CONSTR_02555] [SWS_SwCluC_CONSTR_02556] [SWS_SwCluC_CONSTR_02557] [SWS_SwCluC_CONSTR_02558] [SWS_SwCluC_CONSTR_02559] [SWS_SwCluC_CONSTR_02560] [SWS_SwCluC_CONSTR_02561] [SWS_SwCluC_CONSTR_02562] [SWS_SwCluC_CONSTR_02653] [SWS_SwCluC_CONSTR_02654] [SWS_SwCluC_CONSTR_02655] [SWS_SwCluC_CONSTR_02656] [SWS_SwCluC_CONSTR_02657] [SWS_SwCluC_CONSTR_02658] [SWS_SwCluC_CONSTR_02659] [SWS_SwCluC_CONSTR_02660] [SWS_SwCluC_CONSTR_02664] [SWS_SwCluC_CONSTR_02720] [SWS_SwCluC_CONSTR_02721] [SWS_SwCluC_CONSTR_02752] [SWS_SwCluC_CONSTR_03210] [SWS_SwCluC_CONSTR_03211]

F.2.2 Changed Specification Items in R21-11

[SWS_SwCluC_00005] [SWS_SwCluC_00018] [SWS_SwCluC_00019] [SWS_SwCluC_00031] [SWS_SwCluC_00040] [SWS_SwCluC_00086] [SWS_SwCluC_00999] [SWS_SwCluC_02259] [SWS_SwCluC_02262] [SWS_SwCluC_02263] [SWS_SwCluC_02264] [SWS_SwCluC_03035] [SWS_SwCluC_10009] [SWS_SwCluC_10010] [SWS_SwCluC_10011]

F.2.3 Deleted Specification Items in R21-11

[SWS_NvM_00448] [SWS_NvM_00449] [SWS_NvM_00450] [SWS_NvM_00451] [SWS_NvM_00453] [SWS_NvM_00454] [SWS_NvM_00455] [SWS_NvM_00456] [SWS_NvM_00457] [SWS_NvM_00459] [SWS_NvM_00467] [SWS_NvM_00469] [SWS_NvM_00470] [SWS_NvM_00471] [SWS_NvM_00535] [SWS_NvM_00548] [SWS_NvM_00734] [SWS_NvM_00735] [SWS_NvM_00736] [SWS_NvM_00737] [SWS_NvM_00738] [SWS_NvM_00764] [SWS_NvM_00793] [SWS_NvM_00813] [SWS_NvM_00843] [SWS_NvM_00844] [SWS_NvM_00845] [SWS_NvM_00846] [SWS_NvM_00847] [SWS_NvM_91002] [SWS_Rte_89016] [SWS_Rte_89017] [SWS_Rte_89018] [SWS_Rte_89019] [SWS_Rte_89020] [SWS_Rte_89021] [SWS_Rte_89022] [SWS_Rte_89023] [SWS_Rte_91000] [SWS_Rte_91113] [SWS_Rte_91114] [SWS_Rte_91115] [SWS_Rte_91116] [SWS_Rte_91117] [SWS_Rte_91119]

[SWS_Rte_91121] [SWS_Rte_91122] [SWS_SwCluC_00038] [SWS_SwCluC_00039] [SWS_SwCluC_00071] [SWS_SwCluC_00084] [SWS_SwCluC_02257] [SWS_SwCluC_02258] [SWS_SwCluC_02260] [SWS_SwCluC_02261] [SWS_SwCluC_CONSTR_00074]

F.3 Changes in R22-11 compared to R21-11

F.3.1 Added Specification Items in R22-11

[SWS_SwCluC_00095] [SWS_SwCluC_02278] [SWS_SwCluC_02279] [SWS_SwCluC_02280] [SWS_SwCluC_03083] [SWS_SwCluC_03084] [SWS_SwCluC_03085] [SWS_SwCluC_03086] [SWS_SwCluC_03087] [SWS_SwCluC_03088] [SWS_SwCluC_03089] [SWS_SwCluC_03090] [SWS_SwCluC_03091] [SWS_SwCluC_03092] [SWS_SwCluC_03094] [SWS_SwCluC_03095] [SWS_SwCluC_03096] [SWS_SwCluC_03097] [SWS_SwCluC_03098] [SWS_SwCluC_03099] [SWS_SwCluC_03178] [SWS_SwCluC_03401] [SWS_SwCluC_03402] [SWS_SwCluC_09000] [SWS_SwCluC_09001] [SWS_SwCluC_10012] [SWS_SwCluC_91001] [SWS_SwCluC_91002] [SWS_SwCluC_CONSTR_00096] [SWS_SwCluC_CONSTR_02281] [SWS_SwCluC_CONSTR_02282] [SWS_SwCluC_CONSTR_02283] [SWS_SwCluC_CONSTR_03093] [SWS_SwCluC_CONSTR_03400] [SWS_SwCluC_CONSTR_03403] [SWS_SwCluC_NA_00999]

F.3.2 Changed Specification Items in R22-11

[SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00015] [SWS_SwCluC_00022] [SWS_SwCluC_00027] [SWS_SwCluC_00041] [SWS_SwCluC_00042] [SWS_SwCluC_00043] [SWS_SwCluC_00076] [SWS_SwCluC_00077] [SWS_SwCluC_00085] [SWS_SwCluC_02000] [SWS_SwCluC_02007] [SWS_SwCluC_02101] [SWS_SwCluC_02103] [SWS_SwCluC_02104] [SWS_SwCluC_02105] [SWS_SwCluC_02106] [SWS_SwCluC_02121] [SWS_SwCluC_02125] [SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02143] [SWS_SwCluC_02144] [SWS_SwCluC_02200] [SWS_SwCluC_02201] [SWS_SwCluC_02202] [SWS_SwCluC_02204] [SWS_SwCluC_02205] [SWS_SwCluC_02211] [SWS_SwCluC_02212] [SWS_SwCluC_02213] [SWS_SwCluC_02214] [SWS_SwCluC_02215] [SWS_SwCluC_02216] [SWS_SwCluC_02217] [SWS_SwCluC_02220] [SWS_SwCluC_02221] [SWS_SwCluC_02222] [SWS_SwCluC_02223] [SWS_SwCluC_02224] [SWS_SwCluC_02225] [SWS_SwCluC_02226] [SWS_SwCluC_02227] [SWS_SwCluC_02229] [SWS_SwCluC_02252] [SWS_SwCluC_02253] [SWS_SwCluC_02254] [SWS_SwCluC_02255] [SWS_SwCluC_02256] [SWS_SwCluC_02259] [SWS_SwCluC_02262] [SWS_SwCluC_02263] [SWS_SwCluC_02264] [SWS_SwCluC_02265] [SWS_SwCluC_02266] [SWS_SwCluC_02270] [SWS_SwCluC_02271] [SWS_SwCluC_02301] [SWS_SwCluC_02302] [SWS_SwCluC_02303] [SWS_SwCluC_02308] [SWS_SwCluC_02309] [SWS_SwCluC_02310] [SWS_SwCluC_02311] [SWS_SwCluC_02315] [SWS_SwCluC_02316] [SWS_SwCluC_02317] [SWS_SwCluC_02318]

[SWS_SwCluC_02322] [SWS_SwCluC_02323] [SWS_SwCluC_02324] [SWS_SwCluC_02330] [SWS_SwCluC_02331] [SWS_SwCluC_02333] [SWS_SwCluC_02334] [SWS_SwCluC_02338] [SWS_SwCluC_02339] [SWS_SwCluC_02425] [SWS_SwCluC_02500] [SWS_SwCluC_02501] [SWS_SwCluC_02502] [SWS_SwCluC_02503] [SWS_SwCluC_02504] [SWS_SwCluC_02506] [SWS_SwCluC_02507] [SWS_SwCluC_02510] [SWS_SwCluC_02511] [SWS_SwCluC_02512] [SWS_SwCluC_02513] [SWS_SwCluC_02514] [SWS_SwCluC_02515] [SWS_SwCluC_02516] [SWS_SwCluC_02517] [SWS_SwCluC_02520] [SWS_SwCluC_02533] [SWS_SwCluC_02534] [SWS_SwCluC_02543] [SWS_SwCluC_02548] [SWS_SwCluC_02600] [SWS_SwCluC_02601] [SWS_SwCluC_02602] [SWS_SwCluC_02603] [SWS_SwCluC_02604] [SWS_SwCluC_02605] [SWS_SwCluC_02607] [SWS_SwCluC_02608] [SWS_SwCluC_02609] [SWS_SwCluC_02610] [SWS_SwCluC_02611] [SWS_SwCluC_02612] [SWS_SwCluC_02618] [SWS_SwCluC_02619] [SWS_SwCluC_02620] [SWS_SwCluC_02621] [SWS_SwCluC_02622] [SWS_SwCluC_02623] [SWS_SwCluC_02626] [SWS_SwCluC_02628] [SWS_SwCluC_02629] [SWS_SwCluC_02630] [SWS_SwCluC_02631] [SWS_SwCluC_02632] [SWS_SwCluC_02634] [SWS_SwCluC_02636] [SWS_SwCluC_02637] [SWS_SwCluC_02638] [SWS_SwCluC_02644] [SWS_SwCluC_02645] [SWS_SwCluC_02648] [SWS_SwCluC_02649] [SWS_SwCluC_02652] [SWS_SwCluC_02661] [SWS_SwCluC_02701] [SWS_SwCluC_02702] [SWS_SwCluC_02703] [SWS_SwCluC_02704] [SWS_SwCluC_02705] [SWS_SwCluC_02706] [SWS_SwCluC_02707] [SWS_SwCluC_02723] [SWS_SwCluC_02724] [SWS_SwCluC_02725] [SWS_SwCluC_02726] [SWS_SwCluC_02727] [SWS_SwCluC_02728] [SWS_SwCluC_02736] [SWS_SwCluC_02737] [SWS_SwCluC_02738] [SWS_SwCluC_02739] [SWS_SwCluC_02740] [SWS_SwCluC_02741] [SWS_SwCluC_02742] [SWS_SwCluC_02743] [SWS_SwCluC_02744] [SWS_SwCluC_02746] [SWS_SwCluC_02748] [SWS_SwCluC_02749] [SWS_SwCluC_02751] [SWS_SwCluC_03005] [SWS_SwCluC_03023] [SWS_SwCluC_03024] [SWS_SwCluC_03025] [SWS_SwCluC_03026] [SWS_SwCluC_03027] [SWS_SwCluC_03028] [SWS_SwCluC_03033] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03045] [SWS_SwCluC_03046] [SWS_SwCluC_03047] [SWS_SwCluC_03058] [SWS_SwCluC_03060] [SWS_SwCluC_03061] [SWS_SwCluC_03068] [SWS_SwCluC_03073] [SWS_SwCluC_03074] [SWS_SwCluC_03076] [SWS_SwCluC_03077] [SWS_SwCluC_03101] [SWS_SwCluC_03120] [SWS_SwCluC_03133] [SWS_SwCluC_03142] [SWS_SwCluC_03150] [SWS_SwCluC_03153] [SWS_SwCluC_03156] [SWS_SwCluC_03159] [SWS_SwCluC_03164] [SWS_SwCluC_03165] [SWS_SwCluC_03166] [SWS_SwCluC_03167] [SWS_SwCluC_03168] [SWS_SwCluC_03169] [SWS_SwCluC_03170] [SWS_SwCluC_03171] [SWS_SwCluC_03173] [SWS_SwCluC_03177] [SWS_SwCluC_03201] [SWS_SwCluC_03202] [SWS_SwCluC_03204] [SWS_SwCluC_03209] [SWS_SwCluC_03212] [SWS_SwCluC_03214] [SWS_SwCluC_03215] [SWS_SwCluC_03217] [SWS_SwCluC_03218] [SWS_SwCluC_10000] [SWS_SwCluC_10001] [SWS_SwCluC_10002] [SWS_SwCluC_10003] [SWS_SwCluC_10004] [SWS_SwCluC_10005] [SWS_SwCluC_10006] [SWS_SwCluC_10007] [SWS_SwCluC_10008] [SWS_SwCluC_10009] [SWS_SwCluC_10010] [SWS_SwCluC_10011] [SWS_SwCluC_10020] [SWS_SwCluC_10021]

[SWS_SwCluC_10022] [SWS_SwCluC_10023] [SWS_SwCluC_10032] [SWS_SwCluC_10033] [SWS_SwCluC_10034] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100] [SWS_SwCluC_12200] [SWS_SwCluC_12300] [SWS_SwCluC_12400] [SWS_SwCluC_13000] [SWS_SwCluC_13001] [SWS_SwCluC_13002] [SWS_SwCluC_13003] [SWS_SwCluC_13004] [SWS_SwCluC_13005] [SWS_SwCluC_13006] [SWS_SwCluC_13007] [SWS_SwCluC_13008] [SWS_SwCluC_13009] [SWS_SwCluC_13010] [SWS_SwCluC_13011] [SWS_SwCluC_13012] [SWS_SwCluC_13013] [SWS_SwCluC_13014] [SWS_SwCluC_13015] [SWS_SwCluC_90000] [SWS_SwCluC_90001] [SWS_SwCluC_90002] [SWS_SwCluC_90003] [SWS_SwCluC_90004] [SWS_SwCluC_90005] [SWS_SwCluC_90006] [SWS_SwCluC_90007] [SWS_SwCluC_90008] [SWS_SwCluC_90009] [SWS_SwCluC_90010] [SWS_SwCluC_90011] [SWS_SwCluC_90012] [SWS_SwCluC_90013] [SWS_SwCluC_90014] [SWS_SwCluC_90015] [SWS_SwCluC_90016] [SWS_SwCluC_90017] [SWS_SwCluC_90018] [SWS_SwCluC_90019] [SWS_SwCluC_90020] [SWS_SwCluC_90021] [SWS_SwCluC_90022] [SWS_SwCluC_CONSTR_02135] [SWS_SwCluC_CONSTR_02203] [SWS_SwCluC_CONSTR_02427] [SWS_SwCluC_CONSTR_02546] [SWS_SwCluC_CONSTR_02547] [SWS_SwCluC_CONSTR_02653] [SWS_SwCluC_CONSTR_02659] [SWS_SwCluC_CONSTR_02721] [SWS_SwCluC_CONSTR_03020] [SWS_SwCluC_CONSTR_03021] [SWS_SwCluC_CONSTR_03032] [SWS_SwCluC_CONSTR_03053] [SWS_SwCluC_CONSTR_03059] [SWS_SwCluC_CONSTR_03066] [SWS_SwCluC_CONSTR_03067] [SWS_SwCluC_CONSTR_03211]

F.3.3 Deleted Specification Items in R22-11

[SWS_SwCluC_00999] [SWS_SwCluC_03151] [SWS_SwCluC_03157] [SWS_SwCluC_03174] [SWS_SwCluC_03175] [SWS_SwCluC_CONSTR_02228] [SWS_SwCluC_CONSTR_02229] [SWS_SwCluC_CONSTR_02230]

F.4 Changes in R23-11 compared to R22-11

F.4.1 Added Specification Items in R23-11

[SWS_SwCluC_00097] [SWS_SwCluC_02147] [SWS_SwCluC_02149] [SWS_SwCluC_02150] [SWS_SwCluC_02151] [SWS_SwCluC_02152] [SWS_SwCluC_02153] [SWS_SwCluC_02154] [SWS_SwCluC_02155] [SWS_SwCluC_02156] [SWS_SwCluC_02157] [SWS_SwCluC_03405] [SWS_SwCluC_03406] [SWS_SwCluC_91003]

F.4.2 Changed Specification Items in R23-11

[SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00085] [SWS_SwCluC_02114] [SWS_SwCluC_02140]

[SWS_SwCluC_02426] [SWS_SwCluC_02545] [SWS_SwCluC_02650] [SWS_SwCluC_02750] [SWS_SwCluC_03076] [SWS_SwCluC_03112] [SWS_SwCluC_03113] [SWS_SwCluC_03216] [SWS_SwCluC_09001] [SWS_SwCluC_NA_00999]

F.4.3 Deleted Specification Items in R23-11

none

F.4.4 Added Constraints in R23-11

[SWS_SwCluC_CONSTR_02145] [SWS_SwCluC_CONSTR_02146] [SWS_SwCluC_CONSTR_02148] [SWS_SwCluC_CONSTR_02158] [SWS_SwCluC_CONSTR_02159] [SWS_SwCluC_CONSTR_02160] [SWS_SwCluC_CONSTR_03404]

F.4.5 Changed Constraints in R23-11

none

F.4.6 Deleted Constraints in R23-11

none

F.5 Changes in R24-11 compared to R23-11

F.5.1 Added Specification Items in R24-11

[ECUC_SwCluC_00164] [ECUC_SwCluC_00165]

F.5.2 Changed Specification Items in R24-11

[SWS_SwCluC_02114] [SWS_SwCluC_02125] [SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_10012]

F.5.3 Deleted Specification Items in R24-11

[ECUC_Com_00001] [ECUC_Com_00002] [ECUC_Com_00119] [ECUC_Com_00127] [ECUC_Com_00157] [ECUC_Com_00158] [ECUC_Com_00165] [ECUC_Com_00170] [ECUC_Com_00175] [ECUC_Com_00183] [ECUC_Com_00206]

[ECUC_Com_00259] [ECUC_Com_00263] [ECUC_Com_00314] [ECUC_Com_00340] [ECUC_Com_00344] [ECUC_Com_00345] [ECUC_Com_00387] [ECUC_Com_00391] [ECUC_Com_00412] [ECUC_Com_00437] [ECUC_Com_00493] [ECUC_Com_00518] [ECUC_Com_00519] [ECUC_Com_00520] [ECUC_Com_00560] [ECUC_Com_00709] [ECUC_Com_00711] [ECUC_Com_00761] [ECUC_Com_00765] [ECUC_Com_00811] [ECUC_Com_10003] [ECUC_Com_10006] [ECUC_Com_10010] [ECUC_Com_10011] [ECUC_Com_10012] [ECUC_Com_10014] [ECUC_Com_10017] [ECUC_Com_10018] [ECUC_Com_10019] [ECUC_Com_10020] [ECUC_Com_10021] [ECUC_Com_10025] [ECUC_Com_10026] [ECUC_Com_10027] [ECUC_Com_10028] [ECUC_Com_10029] [ECUC_Com_10030] [ECUC_Com_10031] [ECUC_Com_10032] [ECUC_Com_10033] [ECUC_Com_10034] [ECUC_Com_10035] [ECUC_Com_10036] [ECUC_Com_10038] [ECUC_Com_10039] [ECUC_Com_10040] [ECUC_Dcm_00601] [ECUC_Dcm_00602] [ECUC_Dcm_00604] [ECUC_Dcm_00606] [ECUC_Dcm_00607] [ECUC_Dcm_00611] [ECUC_Dcm_00612] [ECUC_Dcm_00613] [ECUC_Dcm_00614] [ECUC_Dcm_00615] [ECUC_Dcm_00616] [ECUC_Dcm_00617] [ECUC_Dcm_00618] [ECUC_Dcm_00619] [ECUC_Dcm_00620] [ECUC_Dcm_00621] [ECUC_Dcm_00623] [ECUC_Dcm_00624] [ECUC_Dcm_00625] [ECUC_Dcm_00629] [ECUC_Dcm_00640] [ECUC_Dcm_00641] [ECUC_Dcm_00669] [ECUC_Dcm_00670] [ECUC_Dcm_00671] [ECUC_Dcm_00672] [ECUC_Dcm_00673] [ECUC_Dcm_00674] [ECUC_Dcm_00675] [ECUC_Dcm_00676] [ECUC_Dcm_00677] [ECUC_Dcm_00713] [ECUC_Dcm_00720] [ECUC_Dcm_00724] [ECUC_Dcm_00805] [ECUC_Dcm_00807] [ECUC_Dcm_00810] [ECUC_Dcm_00811] [ECUC_Dcm_00824] [ECUC_Dcm_00825] [ECUC_Dcm_00869] [ECUC_Dcm_00894] [ECUC_Dcm_00917] [ECUC_Dcm_00922] [ECUC_Dcm_00923] [ECUC_Dcm_00955] [ECUC_Dcm_00970] [ECUC_Dcm_00985] [ECUC_Dcm_00986] [ECUC_Dcm_00988] [ECUC_Dcm_01012] [ECUC_Dcm_01018] [ECUC_Dcm_01059] [ECUC_Dcm_01060] [ECUC_Dcm_01063] [ECUC_Dcm_01099] [ECUC_Dcm_01106] [ECUC_Dcm_01122] [ECUC_Dcm_01141] [ECUC_Dcm_01142] [ECUC_Dcm_01143] [ECUC_Dcm_01185] [ECUC_Dcm_01215] [ECUC_Dem_00610] [ECUC_Dem_00659] [ECUC_Dem_00660] [ECUC_Dem_00661] [ECUC_Dem_00671] [ECUC_Dem_00702] [ECUC_Dem_00746] [ECUC_Dem_00769] [ECUC_Dem_00792] [ECUC_Dem_00839] [ECUC_Dem_00888] [ECUC_Dem_00894] [ECUC_Dem_00908] [ECUC_Dem_00909] [ECUC_Dem_00916] [ECUC_Dem_00921] [ECUC_Dem_00924] [ECUC_Dem_00929] [ECUC_Dem_00948] [ECUC_Dem_00968] [ECUC_Dem_00969] [ECUC_Dem_00970] [ECUC_EcuC_00001] [ECUC_EcuC_00003] [ECUC_EcuC_00005] [ECUC_EcuC_00006] [ECUC_EcuC_00036] [ECUC_EcuC_00037] [ECUC_EcuC_00052] [ECUC_EcuC_00054] [ECUC_EcuC_00068] [ECUC_EcuC_00072] [ECUC_EcuC_00077] [ECUC_EcuC_00078] [ECUC_EcuC_00079] [ECUC_EcuC_00080] [ECUC_EcuC_00081] [ECUC_EcuC_00082] [ECUC_EcuC_00083] [ECUC_EcuC_00085] [ECUC_EcuC_00086] [ECUC_EcuC_00087] [ECUC_FiM_00039] [ECUC_FiM_00085] [ECUC_LdCom_00001] [ECUC_LdCom_00002] [ECUC_LdCom_00003] [ECUC_LdCom_00005] [ECUC_LdCom_00006] [ECUC_LdCom_00007] [ECUC_LdCom_00010] [ECUC_LdCom_00011] [ECUC_LdCom_00022] [ECUC_LdCom_00023] [ECUC_LdCom_00024] [ECUC_LdCom_00025] [ECUC_LdCom_00026] [ECUC_LdCom_00027] [ECUC_LdCom_00028]

[ECUC_LdCom_00029] [ECUC_LdCom_00030] [ECUC_LdCom_00032] [ECUC_LdCom_00033] [ECUC_LdCom_00034] [ECUC_LdCom_00035] [ECUC_NvM_00028] [ECUC_NvM_00033] [ECUC_NvM_00035] [ECUC_NvM_00036] [ECUC_NvM_00061] [ECUC_NvM_00062] [ECUC_NvM_00072] [ECUC_NvM_00116] [ECUC_NvM_00117] [ECUC_NvM_00119] [ECUC_NvM_00476] [ECUC_NvM_00477] [ECUC_NvM_00478] [ECUC_NvM_00479] [ECUC_NvM_00480] [ECUC_NvM_00481] [ECUC_NvM_00482] [ECUC_NvM_00483] [ECUC_NvM_00484] [ECUC_NvM_00485] [ECUC_NvM_00491] [ECUC_NvM_00492] [ECUC_NvM_00493] [ECUC_NvM_00494] [ECUC_NvM_00495] [ECUC_NvM_00497] [ECUC_NvM_00498] [ECUC_NvM_00499] [ECUC_NvM_00500] [ECUC_NvM_00501] [ECUC_NvM_00502] [ECUC_NvM_00503] [ECUC_NvM_00504] [ECUC_NvM_00505] [ECUC_NvM_00506] [ECUC_NvM_00518] [ECUC_NvM_00519] [ECUC_NvM_00520] [ECUC_NvM_00521] [ECUC_NvM_00532] [ECUC_NvM_00533] [ECUC_NvM_00534] [ECUC_NvM_00538] [ECUC_NvM_00549] [ECUC_NvM_00550] [ECUC_NvM_00551] [ECUC_NvM_00552] [ECUC_NvM_00554] [ECUC_NvM_00555] [ECUC_NvM_00556] [ECUC_NvM_00557] [ECUC_NvM_00558] [ECUC_NvM_00559] [ECUC_NvM_00561] [ECUC_NvM_00562] [ECUC_NvM_00563] [ECUC_NvM_00564] [ECUC_NvM_00565] [ECUC_NvM_00566] [ECUC_NvM_00567] [ECUC_NvM_00572] [ECUC_NvM_00573] [ECUC_Os_00003] [ECUC_Os_00004] [ECUC_Os_00005] [ECUC_Os_00026] [ECUC_Os_00027] [ECUC_Os_00028] [ECUC_Os_00029] [ECUC_Os_00030] [ECUC_Os_00031] [ECUC_Os_00033] [ECUC_Os_00034] [ECUC_Os_00050] [ECUC_Os_00051] [ECUC_Os_00052] [ECUC_Os_00053] [ECUC_Os_00054] [ECUC_Os_00062] [ECUC_Os_00073] [ECUC_Os_00074] [ECUC_Os_00075] [ECUC_Os_00076] [ECUC_Os_00077] [ECUC_Os_00078] [ECUC_Os_00079] [ECUC_Os_00114] [ECUC_Os_00115] [ECUC_Os_00116] [ECUC_Os_00120] [ECUC_Os_00141] [ECUC_Os_00143] [ECUC_Os_00144] [ECUC_Os_00145] [ECUC_Os_00221] [ECUC_Os_00230] [ECUC_Os_00231] [ECUC_Os_00234] [ECUC_Os_00252] [ECUC_Os_00255] [ECUC_Os_00258] [ECUC_Os_00392] [ECUC_Os_00394] [ECUC_Os_00395] [ECUC_Os_00402] [ECUC_Os_00404] [ECUC_Os_01021] [ECUC_Os_01022] [ECUC_Os_01038] [ECUC_Rte_09018] [ECUC_Rte_09019] [ECUC_Rte_09020] [ECUC_Rte_09021] [ECUC_Rte_09023] [ECUC_Rte_09024] [ECUC_Rte_09025] [ECUC_Rte_09026] [ECUC_Rte_09027] [ECUC_Rte_09092] [ECUC_Rte_09115] [ECUC_Rte_09116] [ECUC_Rte_09128] [ECUC_Rte_09129] [ECUC_Rte_09133] [ECUC_Rte_09169] [ECUC_Rte_09170] [ECUC_Rte_09171] [ECUC_Rte_09173] [ECUC_Rte_09214] [ECUC_Rte_89008] [ECUC_SwclC_00009] [ECUC_SwclC_00021] [ECUC_WdgM_00364] [ECUC_WdgM_00366] [ECUC_WdgM_00367] [ECUC_WdgM_00377]

F.5.4 Added Constraints in R24-11

none

F.5.5 Changed Constraints in R24-11

none

F.5.6 Deleted Constraints in R24-11

none

F.6 Changes in R25-11 compared to R24-11

F.6.1 Added Specification Items in R25-11

[ECUC_SwCluC_00166] [SWS_SwCluC_02009] [SWS_SwCluC_02161] [SWS_SwCluC_02162] [SWS_SwCluC_02163] [SWS_SwCluC_02164] [SWS_SwCluC_02284] [SWS_SwCluC_02285] [SWS_SwCluC_02665] [SWS_SwCluC_02666] [SWS_SwCluC_02669] [SWS_SwCluC_02670] [SWS_SwCluC_03502] [SWS_SwCluC_03506] [SWS_SwCluC_03507] [SWS_SwCluC_03508] [SWS_SwCluC_03509] [SWS_SwCluC_03510] [SWS_SwCluC_03511] [SWS_SwCluC_03512] [SWS_SwCluC_03513] [SWS_SwCluC_03514] [SWS_SwCluC_03515] [SWS_SwCluC_03516] [SWS_SwCluC_03517] [SWS_SwCluC_03519] [SWS_SwCluC_03521] [SWS_SwCluC_03522] [SWS_SwCluC_03523] [SWS_SwCluC_03524] [SWS_SwCluC_03525] [SWS_SwCluC_03526] [SWS_SwCluC_03527] [SWS_SwCluC_03528] [SWS_SwCluC_03529] [SWS_SwCluC_03530] [SWS_SwCluC_03531] [SWS_SwCluC_03532] [SWS_SwCluC_03533] [SWS_SwCluC_03534] [SWS_SwCluC_03535] [SWS_SwCluC_03536] [SWS_SwCluC_03537] [SWS_SwCluC_03538] [SWS_SwCluC_03539] [SWS_SwCluC_03540] [SWS_SwCluC_03541] [SWS_SwCluC_03542] [SWS_SwCluC_03543] [SWS_SwCluC_03544] [SWS_SwCluC_03548] [SWS_SwCluC_03549] [SWS_SwCluC_03550] [SWS_SwCluC_03551] [SWS_SwCluC_03552] [SWS_SwCluC_03554] [SWS_SwCluC_91004] [SWS_SwCluC_91005] [SWS_SwCluC_91006] [SWS_SwCluC_91007] [SWS_SwCluC_91008] [SWS_SwCluC_91009] [SWS_SwCluC_91010] [SWS_SwCluC_91011]

F.6.2 Changed Specification Items in R25-11

[ECUC_SwCluC_00012] [ECUC_SwCluC_00017] [ECUC_SwCluC_00036] [ECUC_SwCluC_00044] [ECUC_SwCluC_00053] [ECUC_SwCluC_00056] [ECUC_SwCluC_00061] [ECUC_SwCluC_00062] [ECUC_SwCluC_00063] [ECUC_SwCluC_00064] [ECUC_SwCluC_00076] [ECUC_SwCluC_00085] [ECUC_SwCluC_00087] [ECUC_SwCluC_00107] [ECUC_SwCluC_00114] [ECUC_SwCluC_00120] [ECUC_SwCluC_00127] [ECUC_SwCluC_00131] [ECUC_SwCluC_00147] [SWS_SwCluC_00020] [SWS_SwCluC_02108] [SWS_SwCluC_02265] [SWS_SwCluC_02273] [SWS_SwCluC_02304] [SWS_SwCluC_02305] [SWS_SwCluC_02306] [SWS_SwCluC_02307]

[SWS_SwCluC_02308] [SWS_SwCluC_02312] [SWS_SwCluC_02313] [SWS_SwCluC_02314] [SWS_SwCluC_02315] [SWS_SwCluC_02319] [SWS_SwCluC_02320] [SWS_SwCluC_02321] [SWS_SwCluC_02322] [SWS_SwCluC_02331] [SWS_SwCluC_02332] [SWS_SwCluC_02333] [SWS_SwCluC_02336] [SWS_SwCluC_02337] [SWS_SwCluC_02351] [SWS_SwCluC_02352] [SWS_SwCluC_02354] [SWS_SwCluC_02355] [SWS_SwCluC_02357] [SWS_SwCluC_02358] [SWS_SwCluC_02360] [SWS_SwCluC_02361] [SWS_SwCluC_02363] [SWS_SwCluC_02364] [SWS_SwCluC_02366] [SWS_SwCluC_02367] [SWS_SwCluC_02369] [SWS_SwCluC_02370] [SWS_SwCluC_02372] [SWS_SwCluC_02373] [SWS_SwCluC_02375] [SWS_SwCluC_02376] [SWS_SwCluC_02380] [SWS_SwCluC_02424] [SWS_SwCluC_02604] [SWS_SwCluC_02605] [SWS_SwCluC_02719] [SWS_SwCluC_10012]

F.6.3 Deleted Specification Items in R25-11

[ECUC_SwCluC_00101] [ECUC_SwCluC_00125] [ECUC_SwCluC_00126] [ECUC_SwCluC_00129] [ECUC_SwCluC_00130] [SWS_SwCluC_02218] [SWS_SwCluC_02270] [SWS_SwCluC_02500] [SWS_SwCluC_02501] [SWS_SwCluC_02502] [SWS_SwCluC_02503] [SWS_SwCluC_02504] [SWS_SwCluC_02505] [SWS_SwCluC_02506] [SWS_SwCluC_02507] [SWS_SwCluC_02508] [SWS_SwCluC_02509] [SWS_SwCluC_02510] [SWS_SwCluC_02511] [SWS_SwCluC_02512] [SWS_SwCluC_02513] [SWS_SwCluC_02514] [SWS_SwCluC_02515] [SWS_SwCluC_02516] [SWS_SwCluC_02517] [SWS_SwCluC_02518] [SWS_SwCluC_02519] [SWS_SwCluC_02520] [SWS_SwCluC_02521] [SWS_SwCluC_02522] [SWS_SwCluC_02523] [SWS_SwCluC_02524] [SWS_SwCluC_02525] [SWS_SwCluC_02526] [SWS_SwCluC_02527] [SWS_SwCluC_02528] [SWS_SwCluC_02529] [SWS_SwCluC_02530] [SWS_SwCluC_02531] [SWS_SwCluC_02532] [SWS_SwCluC_02533] [SWS_SwCluC_02534] [SWS_SwCluC_02535] [SWS_SwCluC_02536] [SWS_SwCluC_02537] [SWS_SwCluC_02538] [SWS_SwCluC_02539] [SWS_SwCluC_02540] [SWS_SwCluC_02541] [SWS_SwCluC_02542] [SWS_SwCluC_02543] [SWS_SwCluC_02544] [SWS_SwCluC_02545] [SWS_SwCluC_02548] [SWS_SwCluC_02549] [SWS_SwCluC_02550] [SWS_SwCluC_02551] [SWS_SwCluC_02552] [SWS_SwCluC_13000] [SWS_SwCluC_13008] [SWS_SwCluC_13009] [SWS_SwCluC_13010] [SWS_SwCluC_13011] [SWS_SwCluC_13012] [SWS_SwCluC_13013] [SWS_SwCluC_13014] [SWS_SwCluC_13015] [SWS_SwCluC_90007] [SWS_SwCluC_90012]

F.6.4 Added Constraints in R25-11

[SWS_SwCluC_CONSTR_02667] [SWS_SwCluC_CONSTR_03546] [SWS_SwCluC_CONSTR_03555] [SWS_SwCluC_CONSTR_03556] [SWS_SwCluC_CONSTR_03557] [SWS_SwCluC_CONSTR_03558] [SWS_SwCluC_CONSTR_03559] [SWS_SwCluC_CONSTR_03560] [SWS_SwCluC_CONSTR_03561] [SWS_SwCluC_CONSTR_03562]

F.6.5 Changed Constraints in R25-11

[SWS_SwCluC_CONSTR_03020] [SWS_SwCluC_CONSTR_03053] [SWS_SwCluC_CONSTR_03054]

F.6.6 Deleted Constraints in R25-11

[SWS_SwCluC_CONSTR_02236] [SWS_SwCluC_CONSTR_02546] [SWS_SwCluC_CONSTR_02547] [SWS_SwCluC_CONSTR_02553] [SWS_SwCluC_CONSTR_02554] [SWS_SwCluC_CONSTR_02555] [SWS_SwCluC_CONSTR_02556] [SWS_SwCluC_CONSTR_02557] [SWS_SwCluC_CONSTR_02558] [SWS_SwCluC_CONSTR_02559] [SWS_SwCluC_CONSTR_02560] [SWS_SwCluC_CONSTR_02561] [SWS_SwCluC_CONSTR_02562]

F.7 Changes in R26-11 compared to R25-11

F.7.1 Added Specification Items in R26-11

[ECUC_SwCluC_00167] [SWS_SwCluC_91012]

F.7.2 Changed Specification Items in R26-11

[ECUC_SwCluC_00056] [ECUC_SwCluC_00131] [SWS_SwCluC_02001] [SWS_SwCluC_02132] [SWS_SwCluC_02133] [SWS_SwCluC_10012]

F.7.3 Deleted Specification Items in R26-11

[ECUC_SwCluC_00082] [SWS_SwCluC_02126] [SWS_SwCluC_02615] [SWS_SwCluC_02616] [SWS_SwCluC_02617] [SWS_SwCluC_03509] [SWS_SwCluC_09000] [SWS_SwCluC_91001]

F.7.4 Added Constraints in R26-11

[SWS_SwCluC_CONSTR_02165]

F.7.5 Changed Constraints in R26-11

none

F.7.6 Deleted Constraints in R26-11

none