Document Title	Specification of I-PDU Multiplexer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	182
Document Classification	Standard

Document Version	2.4.1
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	2.4.1	AUTOSAR Release Management	• Editorial changes and minor corrections • No major functional change
31.10.2013	2.4.0	AUTOSAR Release Management	• Revised configuration structure of dynamic and static segments to enforce layout constraints already by the configuration structure • Few bug fixes and clarifications • Editorial changes • Removed chapter(s) on change documentation
19.02.2013	2.3.0	AUTOSAR Administration	• Reworked according to the new SWS_BSWGeneral, harmonization of post-build configuration • Allowing reception of nothing but the static part
02.11.2011	2.2.0	AUTOSAR Administration	• Minor bug fixes and editorial changes • Added configurable JIT-update

Document Change History

Date	Version	Changed by	Change Description
26.10.2010	2.1.0	AUTOSAR Administration	• Updated: tables for mandatory and optional interfaces, SWS_IpduM_00020, SWS_IpduM_00027, SWS_IpduM_00028, SWS_IpduM_00032, SWS_IpduM_00060, SWS_IpduM_00068, SWS_IpduM_00083, SWS_IpduM_00104, ECUC_IpduM_00112, IPDUM117_Conf, SWS_IpduM_00143 and IPDUM162 • Removed: IPDUM013, IPDUM030, IPDUM050_Conf, IPDUM051_Conf, IPDUM063, IPDUM064, IPDUM065, IPDUM072, IPDUM099 and IPDUM154 • Added: pre-compile configuration variant (Chapter 10), ECUC_IpduM_00162, ECUC_IpduM_00163, ECUC_IpduM_00164 and SWS_IpduM_00165
30.11.2009	2.0.0	AUTOSAR Administration	• Harmonization of FIBEX multiplexing and AUTOSAR multiplexing • Many small corrections based on conformance tests and validation activities • Legal disclaimer revised
22.01.2008	1.2.1	AUTOSAR Administration	• Fixed generated figures and captions
31.10.2007	1.2.0	AUTOSAR Administration	• SWS improvements by AUTOSAR Technical Office • Defined maximum I-PDU size for FlexRay to 254 bytes • Document meta information extended • Small layout adaptations made
24.01.2007	1.1.0	AUTOSAR Administration	• Integrated into BSW Scheduler header file structure • Sequence diagrams clarified • Superfluous text removed • Maximum IPDU size clarified • Signature for IpduM_Transmit made consistent with rest of stack. • “Advice for users” revised • Revision Information” added • Legal disclaimer revised

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
12.05.2006	1.0.0	AUTOSAR Administration	Initial release

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, APIs and the configuration of the AUTOSAR Basic Software module I-PDU Multiplexer IpduM.

PDU multiplexing means using the same PCI (Protocol Control Information) of a PDU (Protocol Data Unit) with more than one unique layout of its SDU (Service Data Unit). A selector field is a piece of the SDU of the multiplexed PDU. It is used to distinguish the contents of the multiplexed PDUs from each other.

Multiplexing of PDUs is currently known from CAN, but is not restricted to this communication system.

On sender-side, the I-PDU Multiplexer module is responsible to combine appropriate I-PDUs from COM to new, multiplexed I-PDUs and send them back to the PDU-Router. On receiver-side, it is responsible to interpret the content of multiplexed I-PDUs and provide COM with its appropriate separated I-PDUs taking into account the value of the selector field.

2 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
IpduM	I-PDU Multiplexer
dynamic part	see [4]
static part	see [4]
selector field	see [4]
signal	see [5]
signal group	see [5]
segment	The static or dynamic part may consist of more than one piece. These pieces are called segments. See also SWS_IpduM_00006 and **Figure 3**.
COM I-PDU	I-PDU assembled in the COM module out of COM Signals
IpduM I-PDU	I-PDU assembled in the IpduM module out of two COM I-PDUs
multiplexed I-PDU	see IpduM I-PDU
instance of an I-PDU	IpduM I-PDU with one specific layout and content

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[3] Specification of RTE
AUTOSAR_SWS_RTE.pdf

[4] Requirements on I-PDU Multiplexer
AUTOSAR_SRS_IPDUMultiplexer.pdf

[5] Specification of Communication
AUTOSAR_SWS_COM.pdf

[6] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

None

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [6], which is also valid for IPDU Multiplexer.

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

4 Constraints and assumptions

4.1 Limitations

For transmission of multiplexed I-PDUs, minimum delay time observation cannot be taken into account. For more details, see [5] and 7.4.1.

4.2 Applicability to car domains

No restrictions.

4.3 Applicability to safety related environments

This document has been created in absence of a safety case and a safety plan. Thus, the direct results of this document can only be used within safety relevant systems after repeating certain process steps as required in the IEC 61508.

5 Dependencies to other modules

This chapter lists all the features from other modules that are used by the AUTOSAR IpduM and functionalities that are provided by AUTOSAR IpduM to other modules. Because the IpduM module deals with PDUs that are either sourced or sunk by other modules, care must be taken that shared configuration items are consistent between the modules.

The IpduM is arranged next to the PDU-Router in the layered architecture of AUTOSAR; see [1] and **Figure 1**.

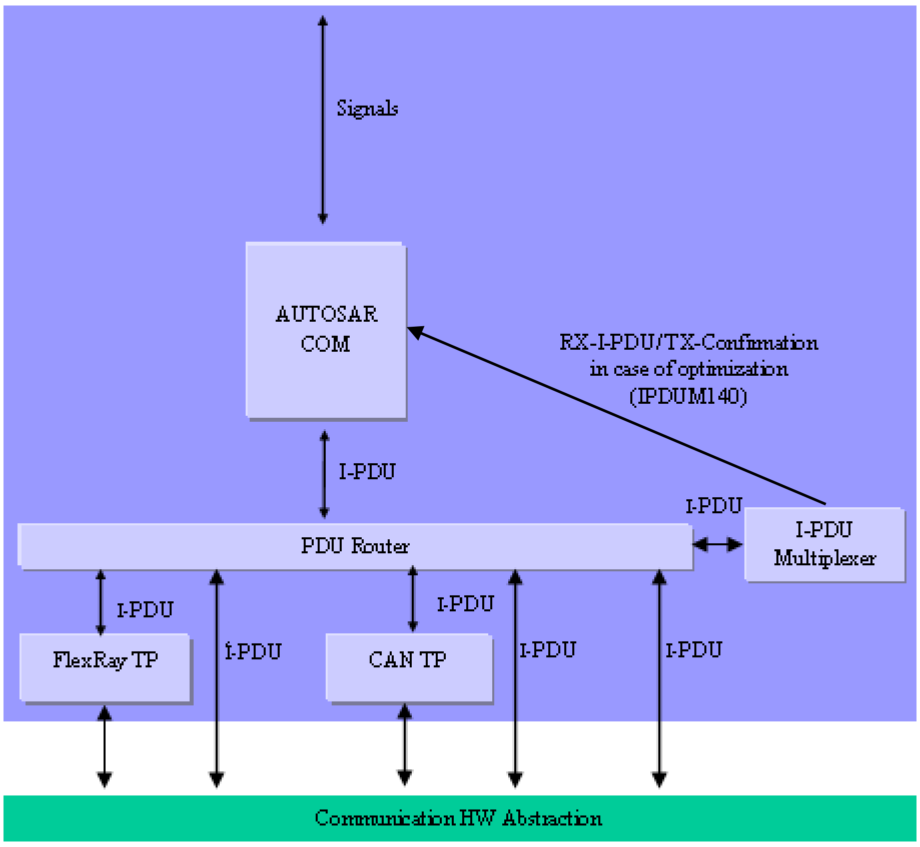

Figure 1 I-PDU Multiplexer in the AUTOSAR Architecture

5.1 AUTOSAR OS

[SWS_IpduM_00107] ⌈The IpduM shall not directly access the AUTOSAR OS. ⌋ (SRS_BSW_00429)

5.2 RTE (BSW Scheduler)

The RTE includes the BSW-Scheduler (see [3]).

The IpduM module relies on the BSW-scheduler calling the IpduM_MainFunction function at a period as configured in IpduMConfigurationTimeBase.

5.3 PDU-Router

The following summarizes the functionality IpduM needs from the PDU-Router (for more details see Chapter 8.6):

- indication of incoming multiplexed I-PDUs
- sending interface for outgoing I-PDUs
- confirmation of I-PDUs which went out

The following list summarizes the functionality provided by the IpduM module for the PDU-Router module:

- indication interface for incoming I-PDUs, which are de-multiplexed
- sending interface for to be multiplexed I-PDUs
- confirmation interface for transmitted I-PDUs

The configuration of the PDU-Router module (e.g. look-up tables) must be such that the I-PDUs, which belong to multiplexed I-PDUs and represent a static or a dynamic part of a multiplexed I-PDU, are routed to the IpduM module.

5.4 COM

The configuration of the IpduM module relies on a corresponding configuration of the AUTOSAR COM module. For each multiplexed I-PDU, there need to be different I-PDUs for the static part and each layout of the dynamic part. For further information configured in the COM module, see Chapter 7.1 and especially **Figure 3**.
The IpduM further assumes that the correct selector field values are already contained in the COM's modules I-PDU representing the dynamic parts. See also SWS_IpduM_00098.

5.5 File structure

5.5.1 Code file structure

This IpduM SWS does not define the code file structure completely.

5.5.2 Header file structure

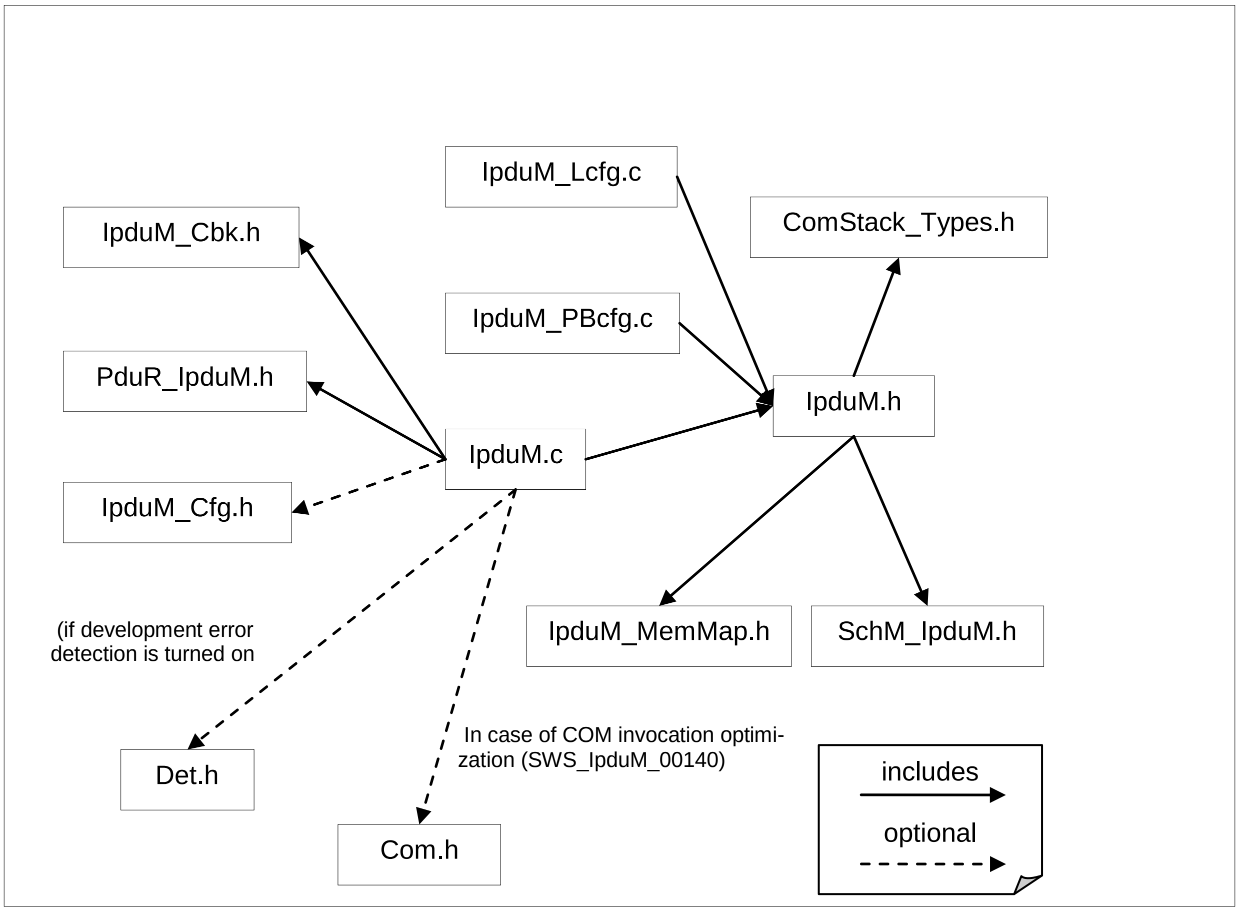

Figure 2 Header File Structure

[SWS_IpduM_00148] ⌈The file IpduM.c shall include IpduM.h, IpduM_Cbk.h, PduR_IpduM.h, and optionally IpduM_Cfg.h, Det.h and Com.h.⌋(SRS_BSW_00415)

[SWS_IpduM_00149] ⌈The file IpduM_Lcfg.c shall include IpduM.h. ⌋ (SRS_BSW_00415)

[SWS_IpduM_00150] 「The file IpduM_PBcfg.c shall include IpduM.h.」(SRS_BSW_00415)

[SWS_IpduM_00151]「File IpduM.h shall include MemMap.h, SchM_IpduM.h and ComStack_Types.h.」(SRS_BSW_00415)

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software [2]

Requirement	Description	Satisfied by
-	-	SWS_IpduM_00104
-	-	SWS_IpduM_00105
-	-	SWS_IpduM_00166
-	-	SWS_IpduM_00168
-	-	SWS_IpduM_00169
-	-	SWS_IpduM_00171
-	-	SWS_IpduM_00172
BSW00431	-	SWS_IpduM_00999
BSW00434	-	SWS_IpduM_00999
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_IpduM_00037
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_IpduM_00999
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hard-ware in a separate initialization function	SWS_IpduM_00032, SWS_IpduM_00033
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_IpduM_00999
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_IpduM_00999
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_IpduM_00999
SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_IpduM_00999
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definiti-on from the service routine	SWS_IpduM_00999
SRS_BSW_00323	All AUTOSAR Basic Software Mo-dules shall check passed API para-meters for validity	SWS_IpduM_00028
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_IpduM_00999

SRS_BSW_00326	-	SWS_IpduM_00999
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_IpduM_00999
SRS_BSW_00338	-	SWS_IpduM_00028
SRS_BSW_00339	Reporting of production relevant error status	SWS_IpduM_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_IpduM_00032
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_IpduM_00102
SRS_BSW_00369	All AUTOSAR Basic Software Mo-dules shall not return specific deve-lopment error codes via the API	SWS_IpduM_00032, SWS_IpduM_00037, SWS_IpduM_00040, SWS_IpduM_00043, SWS_IpduM_00044, SWS_IpduM_00060
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_IpduM_00999
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_IpduM_00999
SRS_BSW_00386	The BSW shall specify the configu-ration for detecting an error	SWS_IpduM_00999
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_IpduM_00032
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_IpduM_00083, SWS_IpduM_00084
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_IpduM_00037
SRS_BSW_00415	Interfaces which are provided exclu-sively for one module shall be sepa-rated into a dedicated header file	SWS_IpduM_00148, SWS_IpduM_00149, SWS_IpduM_00150, SWS_IpduM_00151
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_IpduM_00999
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_IpduM_00999
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_IpduM_00999
SRS_BSW_00425	The BSW module description temp-late shall provide means to model the defined trigger conditions of schedulable objects	SWS_IpduM_00103
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_IpduM_00999
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_IpduM_00107

SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_IpduM_00999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bo-dies provided by the BSW Scheduler	SWS_IpduM_00999
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_IpduM_00999
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_IpduM_00159
SRS_IpduM_02800	For a multiplexed IPDU there shall be exactly one selector field	SWS_IpduM_00004, SWS_IpduM_00007
SRS_IpduM_02801	The size in bits of the selector field shall be configurable	SWS_IpduM_00009
SRS_IpduM_02802	The position of the selector field within the PDU shall be configurable	SWS_IpduM_00005, SWS_IpduM_00155
SRS_IpduM_02803	It shall be possible not to assign a SDU layout to the unused selector field values	SWS_IpduM_00011
SRS_IpduM_02804	For each used selector field value a dynamic and static layout shall be configurable	SWS_IpduM_00006
SRS_IpduM_02806	The three parts of each multiplexed I-PDU must not necessarily be contiguous	SWS_IpduM_00010
SRS_IpduM_02807	The IPDU Multiplexer module shall be designed in a way that it does not produce any additional runtime	SWS_IpduM_00097
SRS_IpduM_02808	It shall be possible that the static part of a IPDU is zero bits long	SWS_IpduM_00004
SRS_IpduM_02809	-	SWS_IpduM_00067, SWS_IpduM_00068, SWS_IpduM_00098, SWS_IpduM_00143
SRS_IpduM_02810	The PduR shall be configured to send parts of multiplexed IPDUs to the IPduM on sender side	SWS_IpduM_00089, SWS_IpduM_00090, SWS_IpduM_00091
SRS_IpduM_02811	-	SWS_IpduM_00021
SRS_IpduM_02812	The PduR shall be configured to send multiplexed IPDUs for de-multiplexing to the IPduM after they were received from the lower layer	SWS_IpduM_00041, SWS_IpduM_00042, SWS_IpduM_00086, SWS_IpduM_00140
SRS_IpduM_02813	The PduR shall be configured to send confirmations related to multi-plexed IPDUs to IPduM after recei-ving them from the lower layer	SWS_IpduM_00022, SWS_IpduM_00101
SRS_IpduM_02814	The confirmation shall depend upon selector field	SWS_IpduM_00019, SWS_IpduM_00020, SWS_IpduM_00023, SWS_IpduM_00024, SWS_IpduM_00087, SWS_IpduM_00088,

		SWS_IpduM_00152
SRS_IpduM_02816	On sender side the IPduM shall combine the static and the appropri-ate dynamic part within IPduM	SWS_IpduM_00015, SWS_IpduM_00017
SRS_IpduM_02817	On receiver side the IPduM extracts the static and dynamic parts of the multiplexed IPDU	SWS_IpduM_00040
SRS_IpduM_02818	The IPduM confirms to COM the static part of the multiplexed IPDU and the dynamic part	SWS_IpduM_00022
SRS_IpduM_02819	There shall be no queuing of trans-mission requests on sender side	SWS_IpduM_00020, SWS_IpduM_00023

Requirement	***Satisfied by***
[SRS_BSW_00344] Reference to link-time configuration	Chapter 10.2.2, SWS_IpduM_00032
[SRS_BSW_00404] Reference to post build time configuration	Chapter 10.2
[SRS_BSW_00405] Reference to multiple configuration sets	SWS_IpduM_00032
[SRS_BSW_00345] Pre-compile-time configuration	Chapter 10.2.2, ECUC_IpduM_00059, ECUC_IpduM_00047, ECUC_IpduM_00048, ECUC_IpduM_00049, ECUC_IpduM_00052, ECUC_IpduM_00053, ECUC_IpduM_00056
[SRS_BSW_00159] Tool-based configuration	not scope of this specification Refers to Configuration WP.
[SRS_BSW_00167] Static configuration check-ing	not scope of this specification Refers to Configuration WP.
[SRS_BSW_00171] Configurability of optional functionality	not applicable (there is no optional functionality)
[SRS_BSW_00170] Data for reconfiguration of AUTOSAR SW-Components	not scope of this specification Refers to Configuration WP.
[SRS_BSW_00380] Separate C-Files for configuration parameters	SWS_IpduM_00095, SWS_IpduM_00096 implementation specific
[SRS_BSW_00419] Separate C-Files for pre-compile time configuration parameters	Chapter 5.5 implementation specific
[SRS_BSW_00381] Separate configuration header file for pre-compile time parameters	Chapter 5.5 implementation specific
[SRS_BSW_00412] Separate H-File for configuration parameters	Chapter 5.5 implementation specific
[SRS_BSW_00383] List dependencies of configuration files	not scope of this specification
[SRS_BSW_00384]	Chapter 5, SWS_IpduM_00104, SWS_IpduM_00105

List dependencies to other modules	
[SRS_BSW_00387] Specify the configuration class of callback function	Chapter 8.5
[SRS_BSW_00388] Introduce containers	Chapter 10.2,ECUC_IpduM_00070, ECUC_IpduM_00071, ECUC_IpduM_00082, ECUC_IpduM_00130
[SRS_BSW_00389] Containers shall have names	Chapter 10.2
[SRS_BSW_00390] Parameter content shall be unique within the module	Chapter 10.2
[SRS_BSW_00391] Parameter shall have unique names	Chapter 10.2
[SRS_BSW_00392] Parameters shall have a type	Chapter 10.2
[SRS_BSW_00393] Parameters shall have a range	Chapter 10.2
[SRS_BSW_00394] Specify the scope of the parameters	Chapter 10.2
[SRS_BSW_00395] List the required parameters (per parameter)	All parameter in Chapter 10.2 are required.
[SRS_BSW_00396] Configuration classes	Chapter 10.2
[SRS_BSW_00397] Pre-compile-time parameters	Chapter 10.2
[SRS_BSW_00398] Link-time parameters	Chapter 10.2
[SRS_BSW_00399] Loadable Post-build time parameters	Chapter 10.2
[SRS_BSW_00400] Selectable Post-build time parameters	Chapter 10.2
[SRS_BSW_00438] Post Build Configuration Data Structure	Chapter 10.2.1, SWS_IpduM_00159
[SRS_BSW_00402] Published information	ECUC_IpduM_00141, ECUC_IpduM_00142, SWS_IpduM_00160
[SRS_BSW_00375] Notification of wake-up reason	not applicable (this layer cannot perform a wake-up)
[SRS_BSW_00101] Initialization interface	SWS_IpduM_00032, SWS_IpduM_00033, SWS_IpduM_00034, SWS_IpduM_00064, SWS_IpduM_00065
[SRS_BSW_00416] Sequence of Initialization	not scope of this specification refere to Mode Management Specification.
[SRS_BSW_00406] Check module initialization	SWS_IpduM_00083, SWS_IpduM_00084
[SRS_BSW_00437] NoInit—Area in RAM	not applicable (not needed)
[SRS_BSW_00168] Diagnostic interface	not applicable (not diagnostic interface included)

[SRS_BSW_00407] Function to read out published parameters	SWS_IpduM_00037
[SRS_BSW_00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	not applicable (this module has no connection to the RTE)
[SRS_BSW_00424] BSW main processing function task allocation	not scope of this specification Implementation specific
[SRS_BSW_00425] Trigger conditions for schedulable objects	SWS_IpduM_00103, ECUC_IpduM_00131
[SRS_BSW_00426] Exclusive areas in BSW modules	not scope of this specification Implementation specific
[SRS_BSW_00427] ISR description for BSW modules	not applicable (module does not provide ISRs)
[SRS_BSW_00428] Execution order dependencies of main processing functions	Chapter 8.6
[SRS_BSW_00429] Restricted BSW OS functionality access	SWS_IpduM_00107
[BSW00431] The BSW Scheduler module implements task bodies	not applicable (requirement for the scheduler)
[SRS_BSW_00432] Modules should have separate main processing functions for read/receive and write/transmit data path	not applicable (transmit and receive functions are called synchronous by the adjacent layers)
[SRS_BSW_00433] Calling of main processing functions	not applicable (requirement for the scheduler)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	not applicable (requirement for the scheduler)
[SRS_BSW_00336] Shutdown interface	not applicable (not needed)
[SRS_BSW_00337] Classification of errors	SWS_IpduM_00026 , SWS_IpduM_00106, SWS_IpduM00153
[SRS_BSW_00338] Detection and Reporting of development errors	SWS_IpduM_00027, SWS_IpduM_00028, ECUC_IpduM_00059, ECUC_IpduM_00132, SWS_IpudM_00154
[SRS_BSW_00369] Do not return development error codes via API	SWS_IpduM_00032, SWS_IpduM_00037, SWS_IpduM_00040, SWS_IpduM_00043, SWS_IpduM_00044, SWS_IpduM_00060
[SRS_BSW_00339] Reporting of production relevant errors and excep-tions	not applicable (module does not define any production relevant errors)
[SRS_BSW_00422] Pre—de—bouncing of production relevant error status	not applicable (not scope of this specification)
[SRS_BSW_00417]	not applicable

Reporting of Error Events by Non-Basic Software	(this module is part of the basic software)
[SRS_BSW_00323] API parameter checking	SWS_IpduM_00028
[SRS_BSW_00004] Version check	SWS_IpduM_00038, SWS_IpduM_00039, ECUC_IpduM_00059, ECUC_IpduM_00134, SWS_IpduM_00165
[SRS_BSW_00409] Header files for production code error IDs	Figure 2
[SRS_BSW_00385] List possible error notifications	SWS_IpduM_00026
[SRS_BSW_00386] Configuration for detecting an error	not applicable (implementation specific)
[SRS_BSW_00161] Microcontroller abstraction	SWS_IpduM_00074, SWS_IpduM_00078
[SRS_BSW_00162] ECU layout abstraction	not applicable (not scope of this specification)
[SRS_BSW_00005] No hard coded horizontal interfaces within MCAL	not applicable (not scope of this specification)
[SRS_BSW_00415] User dependent include files	SWS_IpduM_00148, SWS_IpduM_00149, SWS_IpduM_00150, SWS_IpduM_00151
[SRS_BSW_00164] Implementation of interrupt service routines	not applicable (module does not provide ISRs)
[SRS_BSW_00325] Runtime of interrupt service routines	not applicable (module does not provide ISRs)
[SRS_BSW_00326] Transition from ISRs to OS tasks	not applicable (module does not provide ISRs)
[SRS_BSW_00342] Usage of source code and object code	Chapter 10.2
[SRS_BSW_00343] Specification and configuration of time	Chapter 10.2
[SRS_BSW_00160] Human-readable configuration data	Chapter 10.2
[SRS_BSW_00007] HIS MISRA C	SWS_IpduM_00073
[SRS_BSW_00300] Module naming convention	Figure 2
[SRS_BSW_00413] Accessing instances of BSW modules	not scope of this specification implementation specific
[SRS_BSW_00347] Naming separation of different instances of BSW drivers	not scope of this specification implementation specific
[SRS_BSW_00305] Self-defined data types naming convention	Chapter 8.3.1
[SRS_BSW_00307] Global variables naming	not scope of this specification implementation specific

convention	
[SRS_BSW_00310] API naming convention	Chapter 8.4 and 8.5
[SRS_BSW_00373] Main processing function naming convention	Chapter 8.6
[SRS_BSW_00327] Error values naming convention	SWS_IpduM_00026
[SRS_BSW_00335] Status values naming convention	not scope of this specification implementation specific
[SRS_BSW_00350] Development error detection keyword	SWS_IpduM_00027
[SRS_BSW_00408] Configuration parameter naming convention	Chapter 10.2
[SRS_BSW_00410] Compiler switches shall have defined values	not scope of this specification implementation specific
[SRS_BSW_00411] Get version info keyword	SWS_IpduM_00039
[SRS_BSW_00346] Basic set of module files	Figure 2
[SRS_BSW_00158] Separation of configuration from implementation	Figure 2
[SRS_BSW_00314] Separation of interrupt frames and service routines	not applicable (module does not provide ISRs)
[SRS_BSW_00370] Separation of callback interface from API	Chapter 8.5
[SRS_BSW_00435] Module Header File Structure for the Basic Software Scheduler	Figure 2
[SRS_BSW_00436] Module Header File Structure for the Basic Software Memory Mapping	Figure 2
[SRS_BSW_00348] Standard type header	Figure 2
[SRS_BSW_00353] Platform specific type header	not scope of this specification implementation specific
[SRS_BSW_00361] Compiler specific language extension header	not scope of this specification implementation specific
[SRS_BSW_00301] Limit imported information	not scope of this specification implementation specific
[SRS_BSW_00302] Limit exported information	not scope of this specification implementation specific
[SRS_BSW_00328] Avoid duplication of code	not scope of this specification implementation specific
[SRS_BSW_00312] Shared code shall be reentrant	not scope of this specification implementation specific

[SRS_BSW_00006] Platform independency	not scope of this specification implementation specific
[SRS_BSW_00357] Standard API return type	Chapter 8, SWS_IpduM_00102
[SRS_BSW_00377] Module specific API return types	not applicable (no specific return types)
[SRS_BSW_00304] AUTOSAR integer data types	Figure 2
[SRS_BSW_00355] Do not redefine AUTOSAR integer data types	Chapter 8.3 implementation specific
[SRS_BSW_00378] AUTOSAR boolean type	not scope of this specification implementation specific
[SRS_BSW_00306] Avoid direct use of compiler and platform specific keywords	not scope of this specification implementation specific
[SRS_BSW_00308] Definition of global data	not scope of this specification implementation specific
[SRS_BSW_00309] Global data with read-only constraint	SWS_IpduM_00075, SWS_IpduM_00077
[SRS_BSW_00371] Do not pass function pointers via API	Chapter 8.4 and 8.5
[SRS_BSW_00358] Return type of init functions	Chapter 8.4.1
[SRS_BSW_00414] Parameter of init function	Chapter 8.4.1
[SRS_BSW_00376] Return type and parameters of main processing functions	Chapter 8.6
[SRS_BSW_00359] Return type of callback functions	Chapter 8.5
[SRS_BSW_00360] Parameters of callback functions	Chapter 8.5
[SRS_BSW_00329] Avoidance of generic interfaces	Chapter 8
[SRS_BSW_00330] Usage of macros / inline functions instead of functions	SWS_IpduM_00076, SWS_IpduM_00085
[SRS_BSW_00331] Separation of error and status values	Chapter 8
[SRS_BSW_00009] Module User Documentation	not scope of this specification implementation specific
[SRS_BSW_00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[SRS_BSW_00172]	not scope of this specification

Compatibility and documentation of scheduling strategy	implementation specific
[SRS_BSW_00010] Memory resource documentation	not scope of this specification implementation specific
[SRS_BSW_00333] Documentation of callback function context	not scope of this specification implementation specific
[SRS_BSW_00374] Module vendor identification	Chapter 10.3
[SRS_BSW_00379] Module identification	Chapter 10.3
[SRS_BSW_00003] Version identification	SWS_IpduM_00037, ECUC_IpduM_00059
[SRS_BSW_00318] Format of module version numbers	Chapter 10.3
[SRS_BSW_00321] Enumeration of module version numbers	not scope of this specification implementation specific
[SRS_BSW_00341] Microcontroller compatibility documentation	not scope of this specification implementation specific
[SRS_BSW_00334] Provision of XML file	not scope of this specification Refers to Configuration WP

Document: AUTOSAR requirements on Basic Software cluster IpduM [4]

Requirement	***Satisfied by***
[SRS_IpduM_02800] Exactly one selector field per PDU	SWS_IpduM_00004, SWS_IpduM_00007
[SRS_IpduM_02801] Size of the selector field	SWS_IpduM_00009, ECUC_IpduM_00052
[SRS_IpduM_02802] Position of the selector field	SWS_IpduM_00005, SWS_IpduM_00155
[SRS_IpduM_02815] Compile Time configuration of the selector field	ECUC_IpduM_00052
[SRS_IpduM_02803] Unused values of the selector field	SWS_IpduM_00011
[SRS_IpduM_02804] Support for static and dynamic parts of the PDU	SWS_IpduM_00006
[SRS_IpduM_02808] Support of multiplexed PDUs with a static part of length “zero”	SWS_IpduM_00004, ECUC_IpduM_00133
[SRS_IpduM_02809] Initialization of multiplexed PDUs	SWS_IpduM_00068, SWS_IpduM_00067, SWS_IpduM_00098, SWS_IpduM_00143
[SRS_IpduM_02806] Semantic of the multiplexer	SWS_IpduM_00010
[SRS_IpduM_02810] Routing of multiplexed PDUs on sender side	SWS_IpduM_00089, SWS_IpduM_00090, SWS_IpduM_00091, ECUC_IpduM_00112
[SRS_IpduM_02816] Combining of multiplexed PDUs on sender side	SWS_IpduM_00015, SWS_IpduM_00017, ECUC_IpduM_00114, ECUC_IpduM_00120, ECUC_IpduM_00121, ECUC_IpduM_00125, ECUC_IpduM_00126, ECUC_IpduM_00127, ECUC_IpduM_00128, ECUC_IpduM_00129, ECUC_IpduM_00157
[SRS_IpduM_02811]	SWS_IpduM_00021, ECUC_IpduM_00052

Requirement	***Satisfied by***
Triggering condition on sender side	
[SRS_IpduM_02812] Routing of multiplexed PDUs on receiver side	SWS_IpduM_00041, SWS_IpduM_00042, SWS_IpduM_00086, ECUC_IpduM_00108, ECUC_IpduM_00109, SWS_IpduM_00140
[SRS_IpduM_02817] De-multiplexing PDUs on receiver side	SWS_IpduM_00040, ECUC_IpduM_00113, ECUC_IpduM_00114, ECUC_IpduM_00115
[SRS_IpduM_02813] Routing of Send Confirmations	SWS_IpduM_00022, SWS_IpduM_00101
[SRS_IpduM_02818] Confirmation replication of multiplexed PDUs	SWS_IpduM_00022, ECUC_IpduM_00124, ECUC_IpduM_00163, ECUC_IpduM_00164 , ECUC_IpduM_00158
[SRS_IpduM_02814] Correct confirmation handling of multiplexed PDUs	SWS_IpduM_00023, SWS_IpduM_00024, SWS_IpduM_00019, SWS_IpduM_00020, SWS_IpduM_00087,SWS_IpduM_00088, SWS_IpduM_00152
[SRS_IpduM_02807] No Runtime Overhead for systems without PDU multiplexing	SWS_IpduM_00097
[SRS_IpduM_02819] No queuing of transmission requests on sender side	SWS_IpduM_00020, SWS_IpduM_00023

AUTOSAR Release 4.0 Concept Incorporation

Concept	***Satisfied by***
Debugging concept	SWS_IpduM_00144, SWS_IpduM_00145, SWS_IpduM_00146, SWS_IpduM_00147

7 Functional specification

7.1 Introduction and definitions

I-PDU multiplexing means using the same I-PDU ID transferred from the PDU-Router to the Communication Hardware Abstraction Layer with more than one unique layout of this I-PDU; see also [1].

[SWS_IpduM_00004] ⌈A multiplexed I-PDU consists of a static part and a dynamic part, where the static part consists of zero or more signals or signal groups. The dynamic part consists of the selector field and one or more signals or signal groups; see **Figure 3**. ⌋ (SRS_IpduM_02800, SRS_IpduM_02808)

The dynamic part of an I-PDU is comparable with a union in “C”. With help of the selector field inside the I-PDU, the actual layout of the I-PDU is selected.

[SWS_IpduM_00005] ⌈The position of the static and the dynamic part of the multiplexer shall be arbitrary and has to be configurable per I-PDU; see **Figure 3**, for configuration see Chapter 10.2.2. ⌋ (SRS_IpduM_02802)

[SWS_IpduM_00006] ⌈It shall be possible that the static and the dynamic part consist of more than one element. These elements of the static or dynamic parts are called segments. ⌋ (SRS_IpduM_02804)

[SWS_IpduM_00007] ⌈There shall be only one selector field within one multiplexed I-PDU. ⌋ (SRS_IpduM_02800)

The value of the selector field defines how the content of the dynamic part of the I-PDU will be interpreted.

[SWS_IpduM_00009] ⌈The selector field of one I-PDU shall have a configurable size between one and eight contiguous bits. ⌋ (SRS_IpduM_02801)

[SWS_IpduM_00010] ⌈The position of the selector field within the I-PDU shall be defined by configuration. ⌋ (SRS_IpduM_02806)

The configuration rules for the selector field are defined in Chapter 10.4.1.

Multiplexing of PDUs is currently only known from CAN, but it is not restricted to this communication system.
However, because the module is layered next to the PDU-Router above the interface layer (Communication Hardware Abstraction) in the AUTOSAR layer architecture this feature also could be used with LIN or FlexRay.

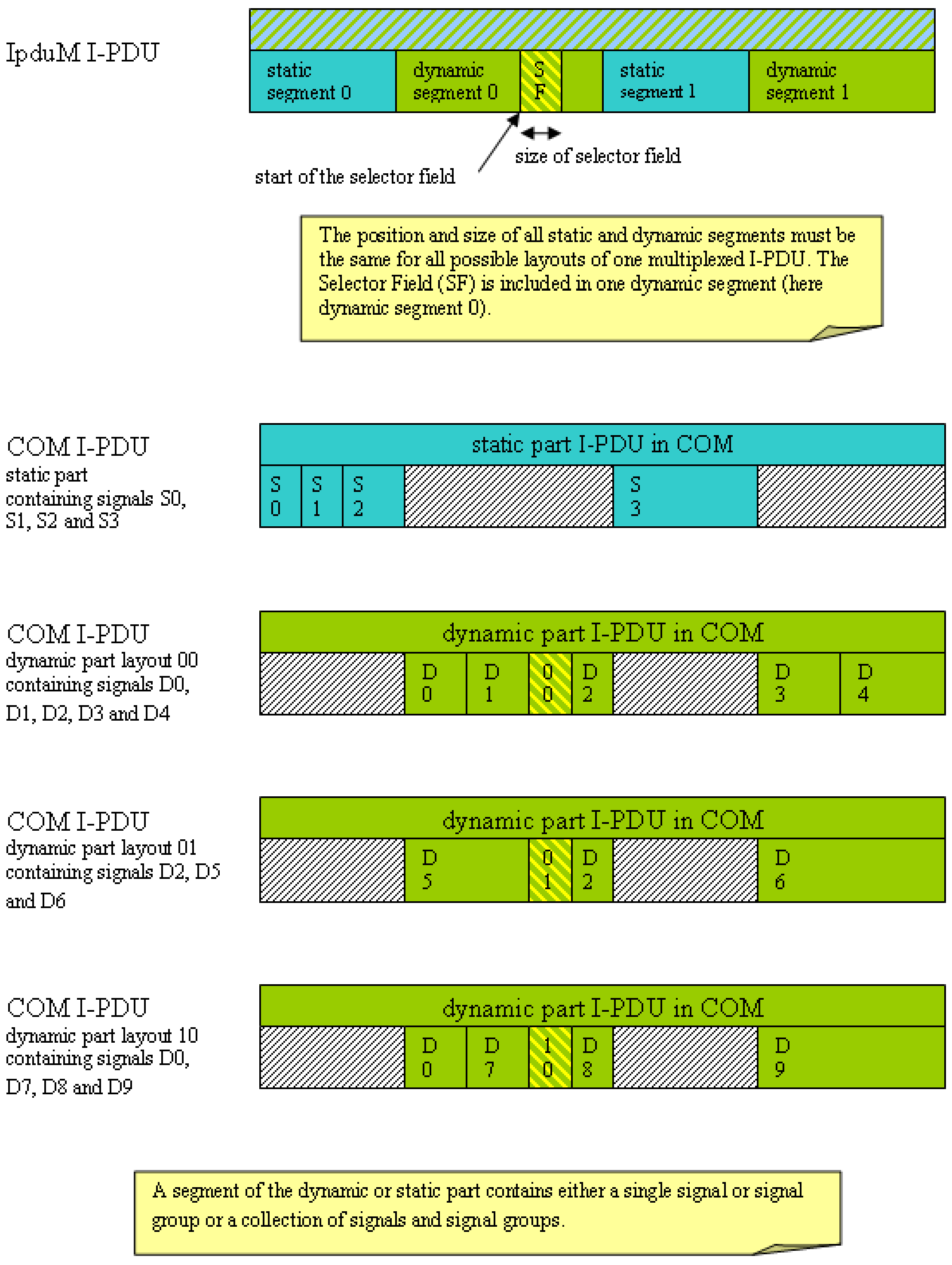
IpduM I-PDU
static segment 0
dynamic segment 0
S F
static segment 1
dynamic segment 1
size of selector field
start of the selector field
The position and size of all static and dynamic segments must be the same for all possible layouts of one multiplexed I-PDU. The Selector Field (SF) is included in one dynamic segment (here dynamic segment 0).
COM I-PDU static part containing signals S0, S1, S2 and S3
static part I-PDU in COM
S 0
S 1
S 2
S 3
COM I-PDU dynamic part layout 00 containing signals D0, D1, D2, D3 and D4
dynamic part I-PDU in COM
D 0
D 1
0 0
D 2
D 3
D 4
COM I-PDU dynamic part layout 01 containing signals D2, D5 and D6
dynamic part I-PDU in COM
D 5
0 1
D 2
D 6
COM I-PDU dynamic part layout 10 containing signals D0, D7, D8 and D9
dynamic part I-PDU in COM
D 0
D 7
1 0
D 8
D 9
A segment of the dynamic or static part contains either a single signal or signal group or a collection of signals and signal groups.

Figure 3 Possible layout of a multiplexed I-PDU

7.2 General

[SWS_IpduM_00097] ⌈The IpduM shall be implemented so that no other modules depend on it and that it is be possible to build a system without the IpduM module if it is not needed. ⌋ (SRS_IpduM_02807)

There is one COM I-PDU for the static part and one COM I-PDU for each layout of the dynamic part of one multiplexed IpduM I-PDU, so the IpduM combines at most two I-PDUs of COM.

[SWS_IpduM_00098] ⌈The IpduM module shall not set the selector field. ⌋ (SRS_IpduM_02809)

The IpduM module relies on the configuration of the COM module. For each dynamic layout, an I-PDU needs to be configured in COM. Such I-PDUs already have to contain the correct selector field value. The selector field values in COM can be initialized by configuring them as signals that are initialized with an init value but are never written after initialization.

For a detailed description of the transmission and reception of a multiplexed I-PDU see Chapter 7.4 and 7.5.

[SWS_IpduM_00140] ⌈It shall be allowed to optimize the Rx- and Tx-Confirmation path from the IpduM module via the PDU-Router module to the COM layer to call the COM API directly from the IpduM module without including the PDU-Router. This shall be indicated by setting the published parameter IpduMRxDirectComInvocation to TRUE, see ECUC_IpduM_00142. ⌋ (SRS_IpduM_02812)

In case of the COM invocation, optimization as defined above IpduM.c needs to include Com.h, see Figure 2 Header File Structure.

7.3 Initialization

The IpduM module provides an initialization function IpduM_Init defined in SWS_IpduM_00032. This function initializes all internal global variables and the buffers of the IpduM I-PDUs. For more details, see Chapter 8.3.1.

The environment of the IpduM shall call IpduM_Init before calling any other function of the IpduM module.

The implementer has to ensure that IPDUM_E_UNINIT is returned in development mode in case an API function is called before the module is initialized.

For the I-PDU data transmission pathway through the IpduM module, a buffer is allocated inside the IpduM module. This buffer needs to be initialized because it might be transmitted before it has been fully populated with data by the COM module. The initialization data of this buffer is derived from the initial values of the COM module's configuration as follows:

1) **[SWS_IpduM_00067]** ⌈The IpduM shall initialize its internal transmit buffers with the configured pattern IpduMIPduUnusedAreasDefault.⌋ (SRS_IpduM_02809)

2) **[SWS_IpduM_00068]** ⌈The initial signal values of the initial dynamic part shall be set according to initial values of the referenced COM I-PDU (IpduMInitialDynamicPart -> IpduMTxDynamicPart -> IpduMTxDynamicPduRef).⌋ (SRS_IpduM_02809)

3) **[SWS_IpduM_00143]** ⌈The initial signal values of the static part shall be set according to the intial values of the referenced COM I-PDU (IpduMTxStaticPart -> IpduMTxStaticPduRef)⌋ (SRS_IpduM_02809)

The selector field is contained within one segment of the intial dynamic part and therefore is initialized implicitly.

For optimization, the initial bit pattern for the buffer can be worked out at configuration-time and then copied at run-time.

7.4 Transmission

Inside COM, there are separated I-PDUs for the static part and one for each dynamic part of a multiplexed I-PDU.

The static part and the dynamic parts are treated in COM as separate I-PDUs with their own I-PDU IDs.

[SWS_IpduM_00015] ⌈For a multiplexed I-PDU IpduM shall merge the corresponding two COM I-PDUs representing the associated static part and the last received dynamic part into one single IpduM I-PDU with a new unique I-PDU ID. IpduM shall send out this new IpduM I-PDU to the PDU-Router module, see also **Figure 1**.⌋ (SRS_IpduM_02816)

For details about the trigger of the transmission, see Chapter 7.4.2.

All control functionalities like deadline monitoring of the COM I-PDUs and update-bit evaluation are out of the scope of the IpduM and have to be done by the COM layer. For details about the timing-behavior of the new combined I-PDU see Chapter 7.4.2.

7.4.1 Transmission request

The IpduM module provides an IpduM_Transmit function so that the PDU-R is able to initiate the transmission of an I-PDU; see SWS_IpduM_00043.

[SWS_IpduM_00017] ⌈The function IpduM_Transmit shall assemble the multiplexed I-PDU, using the related static and dynamic part, and transmit it according to the trigger conditions/ modes as defined in SWS_IpduM_00021 and ECUC_IpduM_00125.⌋ (SRS_IpduM_02816)

As defined in Chapter 7.3, each outgoing I-PDU has an initial value so that, should an I-PDU be transmitted by the IpduM module before both static and dynamic parts have been sent from COM to the IpduM, a value defined by the configuration is transmitted.

[SWS_IpduM_00019] ⌈The configuration of the IpduM shall contain a dedicated timeout for each IpduM I-PDU within the IpduM module in the configuration parameter IpduMTxConfirmationTimeout. ⌋ (SRS_IpduM_02814)

This timeout defines until when the transmission confirmation for this I-PDU has to be received after the transmission. For transmission confirmation, see Chapter 7.4.3.

The timeout period shall take into account the delays in the lower layers.

[SWS_IpduM_00020] ⌈In case the IpduMTxConfirmationTimeout was configured to a value greater than 0, as long as the corresponding timeout timer has not elapsed, and no transmission confirmation for that multiplexed I-PDU was received, the function IpduM_Transmit shall not allow a new transmission request from the upper layer with a COM I-PDU that belongs to the same IpduM I-PDUs.⌋ (SRS_IpduM_02814, SRS_IpduM_02819)

In case IpduMTxConfirmationTimeout was omitted or configured to 0, the IpduM module does not block any new transmission requests.

[SWS_IpduM_00152] ⌈As long as the timeout (defined in the configuration parameter IpduMTxConfirmationTimeout) has not elapsed and as long as no transmission confirmation for the IpduM I-PDU is received, the function IpduM_Transmit shall return with E_NOT_OK for a new transmission request from the upper layer with a COM I-PDU that belongs to the same IpduM I-PDUs. ⌋ (SRS_IpduM_02814)

If the IpduMTxConfirmationTimeout is omitted or configured to 0, the parts of the multiplexed I-PDU may be overwritten even in case they were not already sent or confirmed.

In case a multiplexed I-PDU is only triggered for sending by either updating the dynamic or static part, the non-triggering part might be overwritten if updated multiple times between two transmissions even with a configured IpduMTxConfirmationTimeout. This happens, since the confirmation timeout timer is only started, if the triggering part is updated.

It may be useful to configure the IpduM transmission confirmation timeout depended of the transmission deadline monitoring timeouts for the single COM I-PDUs of the COM layer configuration; see also [5].

7.4.2 Transmission trigger

The IpduM module receives the static and the dynamic part of a multiplexed I-PDU by separated two transmission requests as two single COM I-PDUs from the PDU-Router module.

[SWS_IpduM_00021] ⌈The IpduM module shall be configurable to send a transmission request for the new multiplexed I-PDU to the PDU-Router because of the following trigger conditions/ modes:

- receiving a static part
- receiving a dynamic part
- receiving a static or a dynamic part
- does not trigger transmission because of receiving anything of this I-PDU (IpduMTxTriggerMode None) in case of TriggerTransmit

For configuration, see ECUC_IpduM_00052. ⌋ (SRS_IpduM_02811)

The four trigger conditions/ modes defined by SWS_IpduM_00021 allow controlling the transmission mode of the new assembled I-PDU by the transmission modes of the single I-PDUs sent by COM, see also [5].

Not all of four trigger conditions/ modes defined by SWS_IpduM_00021 guarantee the minimum delay time between consecutive transmissions of different instances of multiplexed I-PDUs, because if the transmission is triggered by static and dynamic part or only by the dynamic part, COM does not take care for the minimum delay time. COM treats the static part and the different dynamic parts as unrelated stand-alone I-PDUs.

The configuration “does not trigger transmission because of receiving anything” is needed if an I-PDU is only sent out because of a TriggerTransmit of a lower layer. With the API IpduM_TriggerTransmit it is possible for lower layers to trigger a send out of an I-PDU.

In case the IpduMTxTriggerMode is None and the lower layer triggers the transmission via IpduM_TriggerTransmit, the IpduMTxConfirmationPduId needs to be configured since this ID is also used for resolving the I-PDU in case of IpduM_TriggerTransmit, see also ECUC_IpduM_00158.

7.4.3 Just-In-Time update of parts

Sometimes it may be unwanted that the IpduM module not just sends out the locally stored parts, since these parts may contain outdated information e.g. update-bits. Therefore, the IpduM supports a per part configurable just-in-time update mechanism.

[SWS_IpduM_00168] ⌈In case the transmission of a multiplexed I-PDU is triggered by the update of one part and IpduMJitUpdate is configured to true for the second part, the IpduM module shall update the second part via PduR_IpduMTriggerTransmit before the multiplexed I-PDU is sent out via PduR_IpduMTransmit. ⌋ ()

[SWS_IpduM_00171] ⌈In case the transmission of a multiplexed I-PDU is triggered by the update of one part and IpduMJitUpdate is configured to true for the second part, the multiplexed I-PDU shall not be send if the JIT-update request via PduR_IpduMTriggerTransmit returns E_NOT_OK.⌋ ()

[SWS_IpduM_00169] ⌈In case the contents of a multiplexed I-PDU is requested via IpduM_TriggerTransmit, the IpduM module shall update all parts which have IpduMJitUpdate configured to true before returning the contents of the multiplexed I-PDU. ⌋ ()

[SWS_IpduM_00172] ⌈ In case the contents of a multiplexed I-PDU is requested via IpduM_TriggerTransmit and IpduMJitUpdate is configured to true for any multiplexed part, IpduM_TriggerTransmit shall return E_NOT_OK if any of the JIT-update requests via PduR_IpduMTriggerTransmit return E_NOT_OK.⌋ ()

7.4.4 Transmission confirmation

Transmission confirmations are given to the IpduM module by the PDU-Router according to the configuration of the I-PDUs in the PDU-Router.

[SWS_IpduM_00022] ⌈If the IpduM receives a TxConfirmation for a specific IpduM I-PDU, it shall translate this confirmation into the corresponding confirmations for the COM I-PDUs, which were contained in the last sent out multiplexed IpduM I-PDU. ⌋ (SRS_IpduM_02813, SRS_IpduM_02818)

Depending on the configuration of IpduMTxDynamicConfirmation (ECUC_IpduM_00163) and IpduMTxStaticConfirmation (ECUC_IpduM_00164), the IpduM will pass zero, one or two confirmations towards COM for one send request. The number of confirmations given to the upper layer does not depend on the IpduMTxTriggerMode.

Examples:

a) If neither IpduMTxDynamicConfirmation nor IpduMTxStaticConfirmation for the corresponding IpduMTxRequest is configured to true, no COM confirmation is generated.
b) If IpduMTxStaticConfirmation is configured to true but and IpduMTxDynamic-Confirmation is configured to false (or vice versa), then only one COM confirmation is generated.
c) If both IpduMTxStaticConfirmation and IpduMTxDynamicConfirmation is configured to true, then two COM confirmations are generated; to the I-PDU representing the static part and the I-PDU representing the dynamic part.

[SWS_IpduM_00023] ⌈If the Tx-Confirmation is not received within the configured timeout IpduMTxConfirmationTimeout the IpduM shall allow new transmission requests for this specific I-PDU after timeout is elapsed. ⌋ (SRS_IpduM_02814, SRS_IpduM_02819)

[SWS_IpduM_00024] ⌈The IpduM shall discard unexpected Tx-Confirmations silently. This may happen if a previously requested transmit request has been timed out, but is confirmed now. ⌋ (SRS_IpduM_02814)

There is no need for an error entry in the case of timeout violation because this is already done in COM, if needed. In the case of a proper configuration of the communication stack, the timeout violation in the IpduM modules occurs at the same time than the Deadline Monitoring violation in the COM module.

7.5 Reception

Every I-PDU which is received by the Communication Hardware Abstraction (CAN Interface, Lin Interface, FlexRay Interface) is given to the PDU-Router. The PDU-Router routes multiplexed I-PDUs to the IpduM module. The IpduM module separately routes the static and dynamic parts of the multiplexed I-PDU to their destinations.

It is known at configuration-time which incoming I-PDU IDs correspond to multiplexed I-PDUs with a static part configured. The I-PDU ID is all that is necessary to work out if there is a static part present.

As all multiplexed I-PDUs contain a dynamic part this part always has to be routed.

There are no requirements to handle or notify wrongly configured parts. Hence, if the received I-PDU contains segments not configured for reception on this ECU, they will be ignored silently. Furthermore, if an I-PDU is configured with a PduLength of 0, it will also be ignored silently, since no meaningful processing can be configured.

This situation might occur in a gateway setting, if a multiplexed I-PDU is always routed onto another bus by the PDU Router, but contains a signal in one dynamic part that must be passed to the application. In this case, the multiplexed PDU would have to be routed to the IpduM as well.

7.6 Error classification

The following errors and exceptions shall be detectable by the IpduM module depending on its build version (development/production mode):

	Type or error	*Relevance*	*Related error code*	*Value [hex]*
SWS_IpduM_00026	API service called with wrong parameter	Development	IPDUM_E_PARAM	10
SWS_IpduM_00162	API service called with a NULL pointer. In case of this error, the API service shall return immediately without any further action, except for reporting this development error.	Development	IPDUM_E_PARAM_POINTER	11
SWS_IpduM_00153	API service used without module initialization	Development	IPDUM_E_UNINIT	20

7.7 Error detection and notification

[SWS_IpduM_00028] ⌈If IpduMDevErrorDetect is configured to TRUE, all IpduM APIs shall check their input parameters and report detected errors to DET by IPDUM_E_PARAM for normal parameter and IPDUM_E_PARAM_POINTER for pointer parameters. ⌋ (SRS_BSW_00338, SRS_BSW_00323)

8 API specification

8.1 Imported types

This chapter lists all imported types and the corresponding header files.

[SWS_IpduM_00102] ⌈

Module	Imported Type
ComStack_Types	PduIdType
	PduInfoType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ (SRS_BSW_00357)

8.2 Type definitions

8.2.1 IpduM_ConfigType

[SWS_IpduM_00159] ⌈

Name:	`IpduM_ConfigType`
Type:	`Structure`
Range:	Implementation specific.
Description:	This is the type of the data structure containing the initialization data for the I-PDU multiplexer.

⌋ (SRS_BSW_00438)

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 IpduM_Init

[SWS_IpduM_00032] ⌈

Service name:	IpduM_Init	
Syntax:	`void IpduM_Init( const IpduM_ConfigType* config )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	config	Implementation specific structure with configuration parameters.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the I-PDU Multiplexer.	

⌋ (SRS_BSW_00344, SRS_BSW_00405, SRS_BSW_00101, SRS_BSW_00369)

[SWS_IpduM_00033] ⌈The function IpduM_Init shall initialize all module-related global variables. ⌋ (SRS_BSW_00101)

[SWS_IpduM_00083] ⌈In case, the configuration parameter IpduMDevErrorDetect equals TRUE: if the parameter config does not reference a valid configuration, the function IpduM_Init shall raise the development error IPDUM_E_PARAM_POINTER. ⌋ (SRS_BSW_00406)

[SWS_IpduM_00084] ⌈The behavior of the IpduM is unspecified until a correct call to IpduM_Init is made. ⌋ (SRS_BSW_00406)

8.3.2 IpduM_GetVersionInfo

[SWS_IpduM_00037] ⌈

Service name:	IpduM_GetVersionInfo	
Syntax:	`void IpduM_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Service returns the version information of this module.	

⌋ (SRS_BSW_00407, SRS_BSW_00369, SRS_BSW_00003)

8.3.3 IpduM_Transmit

[SWS_IpduM_00043] ⌈

Service name:	IpduM_Transmit	

Syntax:	`Std_ReturnType IpduM_Transmit(`
	`    PduIdType PdumTxPduId,`
	`    const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.	
Parameters (in):	PdumTxPduId	ID of I-PDU to be transmitted. Range: 0..(maximum number of I-PDU IDs which are multiplexed) - 1
	PduInfoPtr	A pointer to a structure with I-PDU related data that shall be transmitted: data length and pointer to I-SDU buffer
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Transmit request is accepted E_NOT_OK: Transmit request is not accepted
Description:	Service is called by the PDU-Router to request a transmission.	

⌋ (SRS_BSW_00369)

For a detailed description read Chapter 7.4.1.

8.4 Call-back notifications

8.4.1 IpduM_RxIndication

[SWS_IpduM_00040] ⌈

Service name:	IpduM_RxIndication	
Syntax:	`void IpduM_RxIndication( PduIdType RxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x42	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	ID of the received I-PDU.
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication interface module.	

⌋ (SRS_BSW_00369; SRS_IpduM_02817)

[SWS_IpduM_00041] ⌈If there is a static part configured in a multiplexed SDU received from the PDU-R, the function IpduM_RxIndication transforms the incoming I-PDU ID into the correct I-PDU ID for the static part's destination and then forwards the SDU via the PDU-R, see PduR_IpduMRxIndication in the PDU-R SWS.⌋ (SRS_IpduM_02812)

[SWS_IpduM_00042] ⌈When a multiplexed I-PDU is received from the PDU-R the function IpduM_RxIndication uses the incoming I-PDU ID and the selector field to find out the correct I-PDU ID for the dynamic part's destination and then forwards the I-PDU via the PDU-R, see PduR_IpduMRxIndication in the PDU-R SWS.⌋ (SRS_IpduM_02812)

[SWS_IpduM_00086] ⌈The function IpduM_RxIndication shall be callable in interrupt context, e.g. from receive interrupt. ⌋ (SRS_IpduM_02812)

8.4.2 IpduM_TxConfirmation

[SWS_IpduM_00044] ⌈

Service name:	IpduM_TxConfirmation	
Syntax:	`void IpduM_TxConfirmation( PduIdType TxPduId )`	
Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the I-PDU that has been transmitted.
Parameters (inout):	None	

Parameters (out):	None
Return value:	None
Description:	The lower layer communication interface module confirms the transmission of an I-PDU.

⌋ (SRS_BSW_00369)

[SWS_IpduM_00088] ⌈The function IpduM_TxConfirmation shall translate the confirmation received from the PDU-Router into confirmations for the I-PDUs which where contained in the sent multiplexed I-PDU. ⌋ (SRS_IpduM_02814)

These confirmations are given again to the PDU-Router that has to route them to COM.

[SWS_IpduM_00087] ⌈The function IpduM_TxConfirmation shall be callable in interrupt context, e.g. from a transmit interrupt. ⌋ (SRS_IpduM_02814)

8.4.3 IpduM_TriggerTransmit

[SWS_IpduM_00060] ⌈

Service name:	IpduM_TriggerTransmit	
Syntax:	`Std_ReturnType IpduM_TriggerTransmit( PduIdType TxPduId, PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x41	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied. 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 copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength.	

⌋ (SRS_BSW_00369)

[SWS_IpduM_00090] ⌈The function IpduM_TriggerTransmit shall copy the contents of its I-PDU transmit buffer to the I-PDU buffer given by PduInfoPtr.⌋ (SRS_IpduM_02810)

[SWS_IpduM_00091] ⌈The IpduM shall take care about the data consistency during providing the data. ⌋ (SRS_IpduM_02810)

Use case: This function is used e.g. by the LIN Master for sending out a LIN frame. In this case, the trigger transmit can be initiated by the Master schedule table itself or a received LIN header.
This function is also used by the FlexRay Interface for requesting PDUs to be sent in static part (synchronous to the FlexRay global time).

[SWS_IpduM_00089] ⌈The function IpduM_TriggerTransmit shall be callable in interrupt context. ⌋ (SRS_IpduM_02810)

8.5 Scheduled functions

Most of the functions of the IpduM module are called synchronous in the context of the upper layer (for transmission) and in the context of the lower layer (for reception). However, for the TxConfirmation timeout timer a scheduled function is needed.

[SWS_IpduM_00103] ⌈

Service name:	IpduM_MainFunction

Syntax:	`void IpduM_MainFunction(`
	`    void`
	`)`

Service ID[hex]:	0x10
Description:	Performs the processes of the activities that are not directly initiated by the calls from PDU-R.

⌋ (SRS_BSW_00425)

[SWS_IpduM_00101] ⌈The function IpduM_MainFunction shall perform the processing of the IpduM activities that are not directly initiated by the calls from PDU-R. This includes at least the TxConfirmation time observation. ⌋ (SRS_IpduM_02813)

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

This chapter defines all interfaces that are required to fulfill the core functionality of the module.

[SWS_IpduM_00104] ⌈

API function	Description

Actually, the IpduM module needs no APIs of other modules compulsorily, since the IpduM module could be used only for reception or transmission of multiplexed I-PDUs. In such a case the not used reception or transmission APIs of the PduR are optional. Hence, depending on the use-case all used APIs are optional.⌋ ()

8.6.2 Optional Interfaces

This chapter defines all interfaces that are required to fulfill an optional functionality of the module.

[SWS_IpduM_00105] ⌈

API function	Description
Det_ReportError	Service to report development errors.
PduR_IpduMTransmit	Requests transmission of an I-PDU.
PduR_IpduMRxIndication	Indication of a received I-PDU from a lower layer communication interface module.
PduR_IpduMTriggerTransmit	Within this API, the upper layer module (called module) shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength.
PduR_IpduMTxConfirmation	The lower layer communication interface module confirms the transmission of an I-PDU.

⌋ ()

8.6.3 Configurable interfaces

Not applicable

9 Sequence diagrams

9.1 Transmission of a multiplexed I-PDU and Transmit confirmation

The following sequence chart shows a transmit request initiated by the COM layer. The transmit request is for an I-PDU which has to be transmitted within a multiplexed I-PDU. In the IpduM module is configured that this transmitted I-PDU triggers the sending of the multiplexed I-PDU.

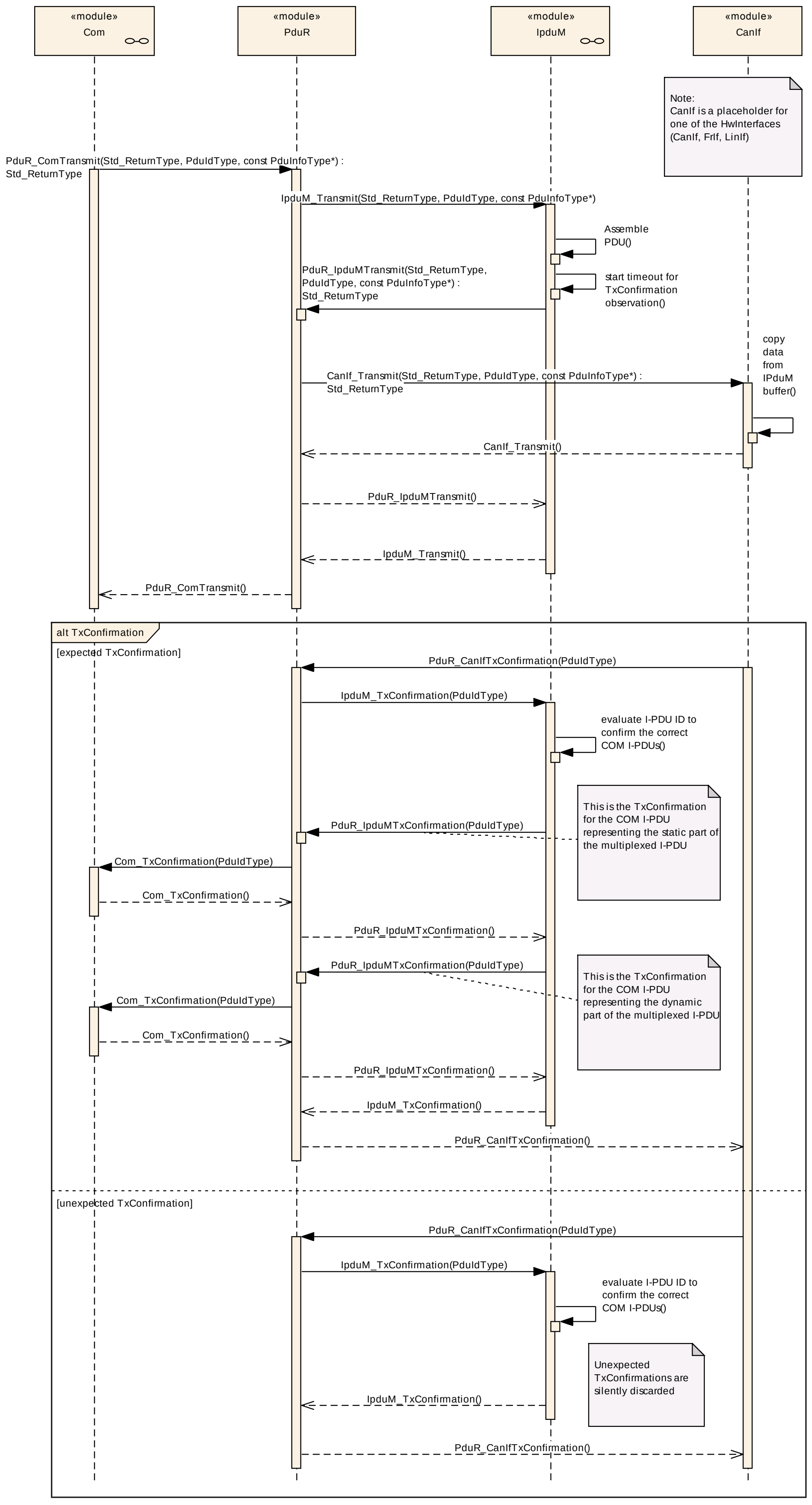
«module» Com
«module» PduR
«module» IpduM
«module» CanIf
Note: CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Assemble PDU()
start timeout for TxConfirmation observation()
PduR_IpduMTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
copy data from IPduM buffer()
CanIf_Transmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
CanIf_Transmit()
PduR_IpduMTransmit()
IpduM_Transmit()
PduR_ComTransmit()
alt TxConfirmation
[expected TxConfirmation]
PduR_CanIfTxConfirmation(PduIdType)
IpduM_TxConfirmation(PduIdType)
evaluate I-PDU ID to confirm the correct COM I-PDUs()
This is the TxConfirmation for the COM I-PDU representing the static part of the multiplexed I-PDU
PduR_IpduMTxConfirmation(PduIdType)
Com_TxConfirmation(PduIdType)
Com_TxConfirmation()
PduR_IpduMTxConfirmation()
PduR_IpduMTxConfirmation(PduIdType)
This is the TxConfirmation for the COM I-PDU representing the dynamic part of the multiplexed I-PDU
Com_TxConfirmation(PduIdType)
Com_TxConfirmation()
PduR_IpduMTxConfirmation()
IpduM_TxConfirmation()
PduR_CanIfTxConfirmation()
[unexpected TxConfirmation]
PduR_CanIfTxConfirmation(PduIdType)
IpduM_TxConfirmation(PduIdType)
evaluate I-PDU ID to confirm the correct COM I-PDUs()
Unexpected TxConfirmations are silently discarded
IpduM_TxConfirmation()
PduR_CanIfTxConfirmation()

Figure 4 Transmission and confirmation of multiplexed I-PDU with triggering

9.2 Transmission of a multiplexed I-PDU without Trigger

The following sequence chart shows a transmit request initiated by the COM layer. Because of the configuration of the IpduM, no transmit request for the IpduM I-PDU takes place. For configuration see ECUC_IpudM_00052.

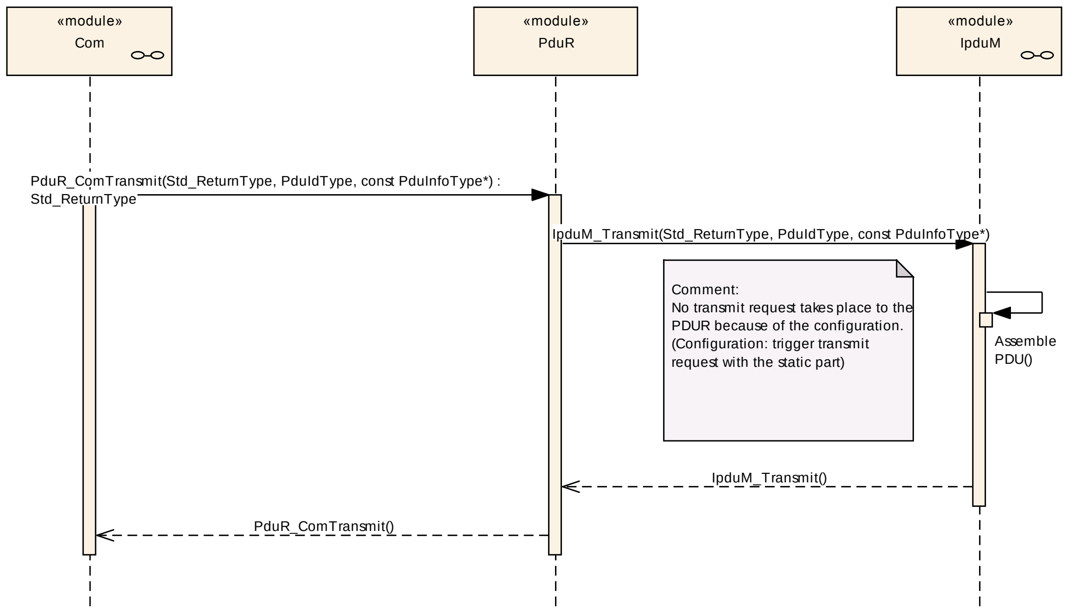
«module»
Com
«module»
PduR
«module»
IpduM
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*) :
Std_ReturnType
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Comment:
No transmit request takes place to the
PDUR because of the configuration.
(Configuration: trigger transmit
request with the static part)
Assemble
PDU()
IpduM_Transmit()
PduR_ComTransmit()

Figure 5 Transmission of a multiplexed I-PDU without triggering

9.3 Reception of the multiplexed I-PDU

The following sequence chart shows a reception of a multiplexed I-PDU. The I-PDU contains a static and a dynamic part and both are configured to create an RxIndication to the PDU-R module.

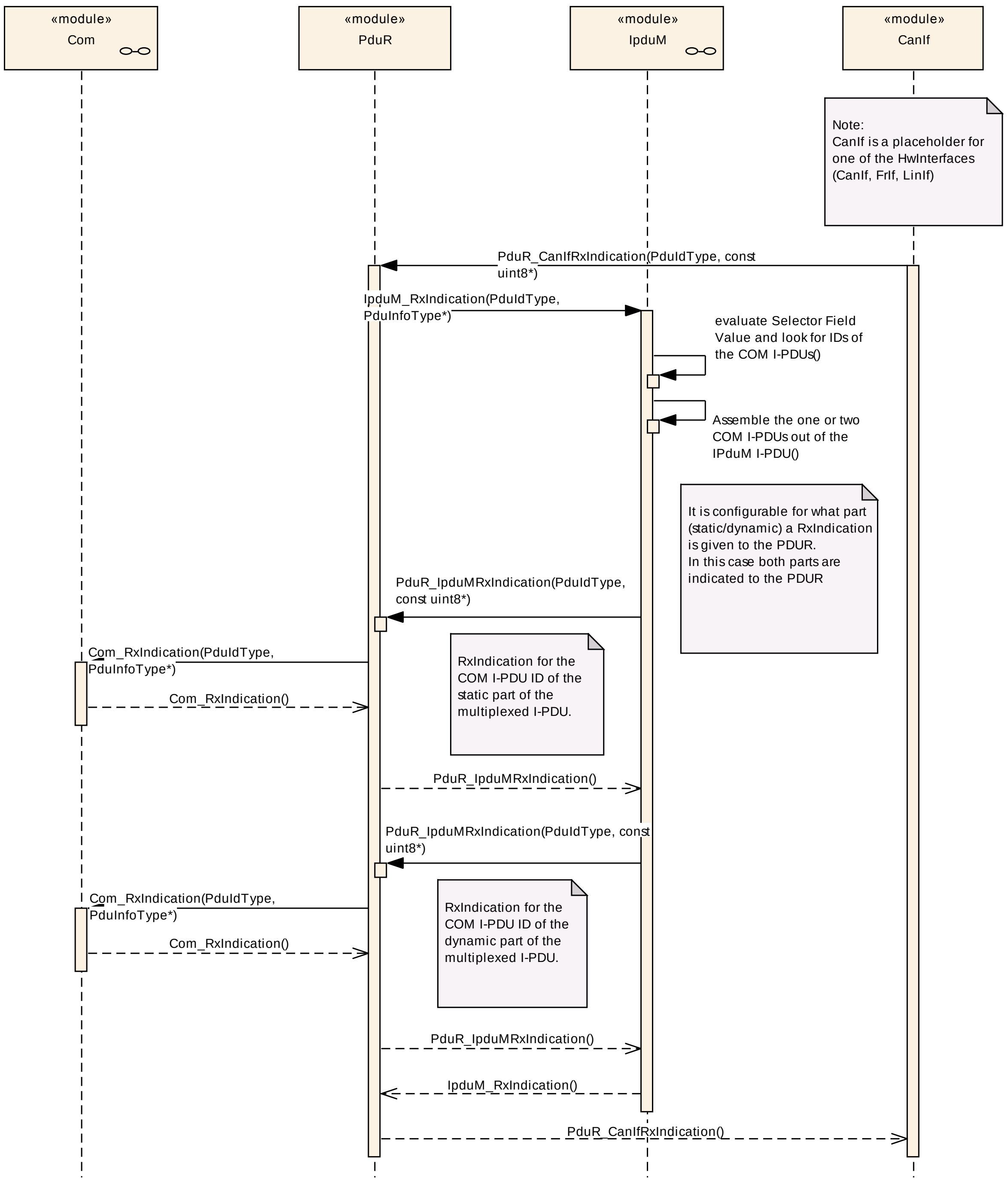
«module» Com
«module» PduR
«module» IpduM
«module» CanIf
Note: CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
PduR_CanIfRxIndication(PduIdType, const uint8*)
IpduM_RxIndication(PduIdType, PduInfoType*)
evaluate Selector Field Value and look for IDs of the COM I-PDUs()
Assemble the one or two COM I-PDUs out of the IPduM I-PDU()
It is configurable for what part (static/dynamic) a RxIndication is given to the PDUR. In this case both parts are indicated to the PDUR
PduR_IpduMRxIndication(PduIdType, const uint8*)
Com_RxIndication(PduIdType, PduInfoType*)
Com_RxIndication()
RxIndication for the COM I-PDU ID of the static part of the multiplexed I-PDU.
PduR_IpduMRxIndication()
PduR_IpduMRxIndication(PduIdType, const uint8*)
Com_RxIndication(PduIdType, PduInfoType*)
Com_RxIndication()
RxIndication for the COM I-PDU ID of the dynamic part of the multiplexed I-PDU.
PduR_IpduMRxIndication()
IpduM_RxIndication()
PduR_CanIfRxIndication()

Figure 6 Reception of a multiplexed I-PDU

9.4 Trigger Transmit

The following sequence chart shows a Trigger Transmit request from an interface layer.

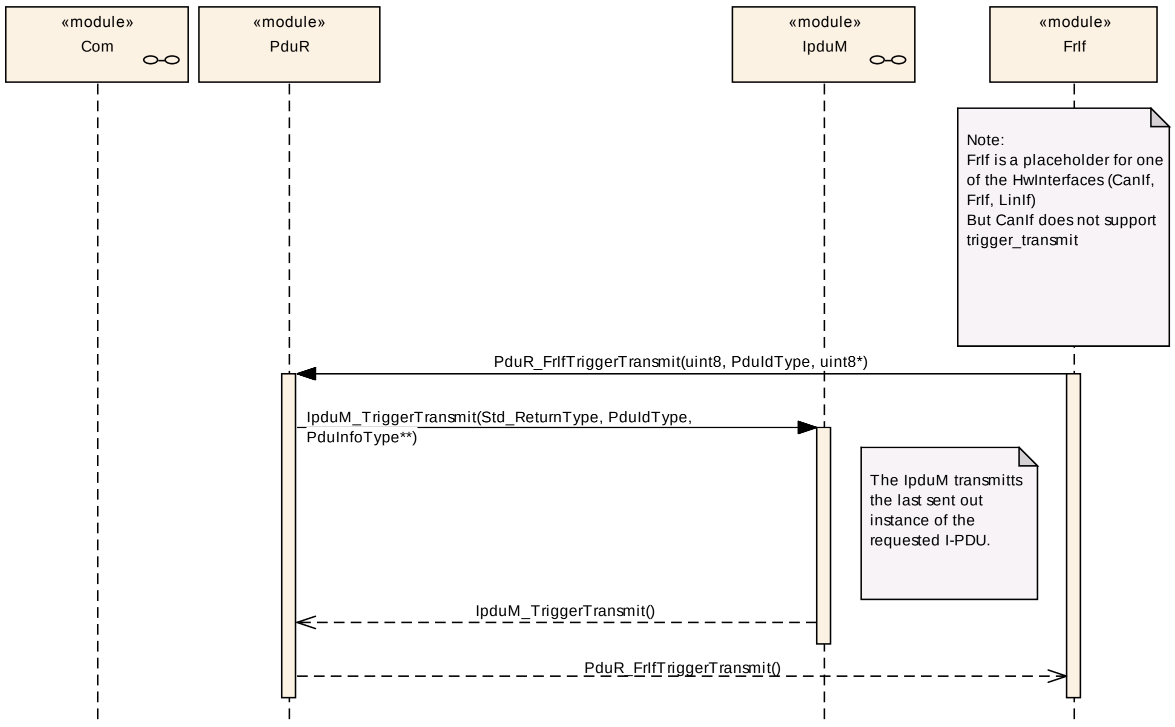
«module»
Com
«module»
PduR
«module»
IpduM
«module»
FrIf
Note:
FrIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
But CanIf does not support trigger_transmit
PduR_FrIfTriggerTransmit(uint8, PduIdType, uint8*)
IpduM_TriggerTransmit(Std_ReturnType, PduIdType, PduInfoType**)
The IpduM transmitts the last sent out instance of the requested I-PDU.
IpduM_TriggerTransmit()
PduR_FrIfTriggerTransmit()

Figure 7 Trigger Transmit request from interface layer

9.5 Missing Transmit Confirmation

The following sequence chart shows the case that a TxConfirmation is not received by the IpduM module during the TX Confirmation timeout. After the timeout has elapsed, it is allowed to send the I-PDU again.

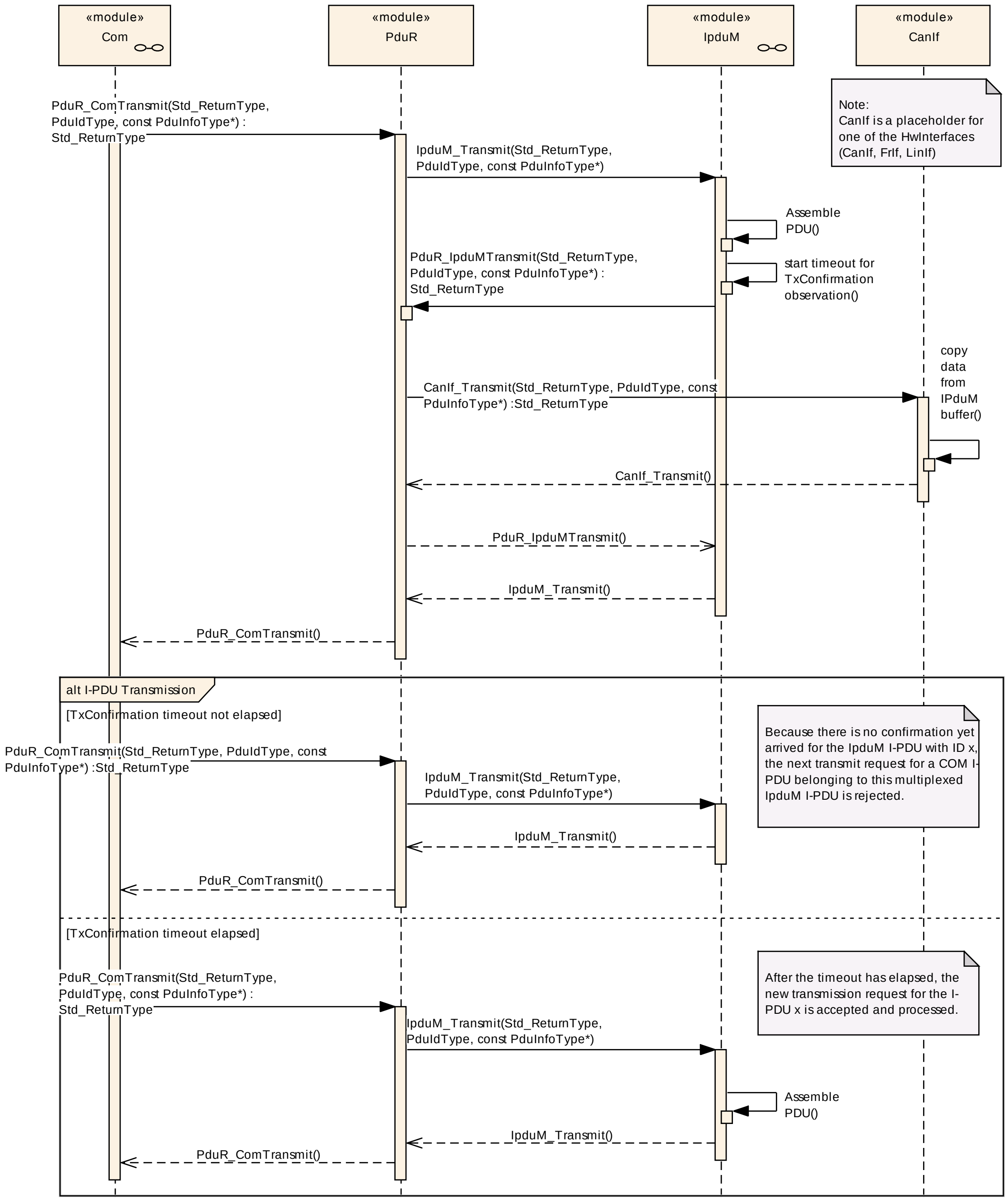
«module» Com
«module» PduR
«module» IpduM
«module» CanIf
Note: CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Assemble PDU()
start timeout for TxConfirmation observation()
PduR_IpduMTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
CanIf_Transmit(Std_ReturnType, PduIdType, const PduInfoType*) :Std_ReturnType
copy data from IPduM buffer()
CanIf_Transmit()
PduR_IpduMTransmit()
IpduM_Transmit()
PduR_ComTransmit()
alt I-PDU Transmission
[TxConfirmation timeout not elapsed]
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*) :Std_ReturnType
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Because there is no confirmation yet arrived for the IpduM I-PDU with ID x, the next transmit request for a COM I-PDU belonging to this multiplexed IpduM I-PDU is rejected.
IpduM_Transmit()
PduR_ComTransmit()
[TxConfirmation timeout elapsed]
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
After the timeout has elapsed, the new transmission request for the I-PDU x is accepted and processed.
Assemble PDU()
IpduM_Transmit()
PduR_ComTransmit()

Figure 8 Missing Transmit Confirmation

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers.

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

Chapter 10.3 specifies published information of the module IpduM.

10.1 How to read this chapter

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

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

10.2.1 Variants

There are three variants called: VARIANT-PRE-COMPILE, VARIANT-LINK-TIME and VARIANT-POST-BUILD.

The VARIANT-PRE_COMPILE is designed for modules that are purely configured at pre-compile time. In this variant, all configuration parameters are fixed at compile-time.

The VARIANT-LINK-TIME is designed for the use case where parameters that affect code generation are fixed at compile-time and all other configuration parameters are fixed at link-time.

The VARIANT-POST-BUILD is designed for parameters that affect code generation to be fixed at compile-time and all other parameters to be fixed at post build-time.

10.2.2 Configuration overview

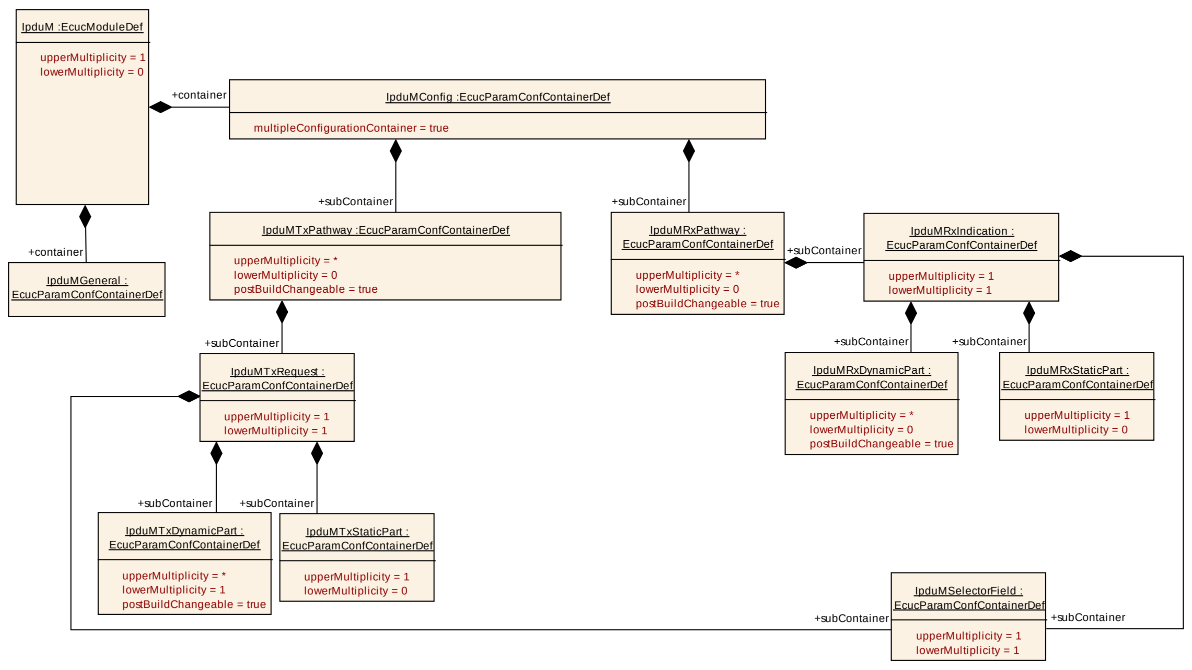
IpduM :EcucModuleDef
upperMultiplicity = 1
lowerMultiplicity = 0
+container
IpduMConfig :EcucParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
+subContainer
+container
IpduMGeneral : EcucParamConfContainerDef
IpduMTxPathway :EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
postBuildChangeable = true
IpduMRxPathway : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
postBuildChangeable = true
+subContainer
IpduMRxIndication : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+subContainer
IpduMTxRequest : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+subContainer
+subContainer
IpduMRxDynamicPart : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
postBuildChangeable = true
IpduMRxStaticPart : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
+subContainer
+subContainer
IpduMTxDynamicPart : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
postBuildChangeable = true
IpduMTxStaticPart : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 0
IpduMSelectorField : EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+subContainer
+subContainer

Figure 9 IpduM Configuration Overview

10.2.3 IpduM

Module Name	**IpduM**
Module Description	Configuration of the IpduM (Ipdu Multiplexer) module.

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMConfig	1	This container contains the sub containers of the IpduM module. The IpduMTxPathway subcontainer includes information about sent I-PDUs. The IpduMRxPathway includes information about received I-PDUs. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
IpduMGeneral	1	Contains the general configuration parameters of IpduM.
IpduMPublishedInformation	1	Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, since they are published information.

10.2.4 IpduMConfig

SWS Item	**ECUC_IpduM_00059 :**
Container Name	IpduMConfig [Multi Config Container]
Description	This container contains the sub containers of the IpduM module. The IpduMTxPathway subcontainer includes information about sent I-PDUs. The IpduMRxPathway includes information about received I-PDUs. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00166 :**		
Name	IpduMMaxTxBufferSize		
Description	Maximum total size of all Tx buffers. This parameter is needed only in case of post-build loadable implementation using static memory allocation.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00165 :**
Name	IpduMMaxTxPathwayCnt
Description	Maximum number of transmitted IPdus. This parameter is needed only in case of post-build loadable implementation using static memory allocation.
Multiplicity	0..1

Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMRxPathway	0..*	includes information about received I-PDUs
IpduMTxPathway	0..*	includes information about sent I-PDUs

10.2.5 IpduMGeneral

SWS Item	**ECUC_IpduM_00130 :**
Container Name	IpduMGeneral
Description	Contains the general configuration parameters of IpduM.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00131 :**		
Name	IpduMConfigurationTimeBase		
Description	The cycle time with which IpduM_MainFunction should be invoked (in seconds).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 3600		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00132 :**		
Name	IpduMDevErrorDetect		
Description	Active/Deactivate the detection of development errors, for production code this parameter has to be False. True: error detection activated False: error detection deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00133 :**
Name	IpduMStaticPartExists
Description	This is to allow optimizations in the case the IpduM will never be used with a static part. Note that this is a pre-compile option. If this is set to False then it will not

	be possible to add static parts after compilation. True: A static part may exist. False: A static part will never exist.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00134 :**		
Name	IpduMVersionInfoApi		
Description	Active/Deactivate the version information API. true: version information activated false: version information deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 IpduMTxPathway

SWS Item	**ECUC_IpduM_00070 :**
Container Name	IpduMTxPathway
Description	Contains the configuration parameters transmitted I-PDUs by the IpduM module. **Attributes:** postBuildChangeable=true
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMTxRequest	1	configuration for a TxRequest

10.2.7 IpduMTxRequest

SWS Item	**ECUC_IpduM_00052 :**
Container Name	IpduMTxRequest
Description	This container is used to specify the configuration for Transmit requests. There will be one instance of this container for each I-PDU that can be requested for transmission (the outgoing I-PDUs) by the IpduM.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00162 :**		
Name	IpduMByteOrder		
Description	This parameter defines the ByteOrder for all segments (static and dynamic part) and for the selectorField within the MultiplexedPdu. The absolute position of a segment in the MultiplexedIPdu is determined by the definition of the ByteOrder parameter: If BIG_ENDIAN is specified, the Seg-mentPosition indicates the bit position of the most significant bit in an IPDU. If LITTLE_ENDIAN is specified, the SegmentPosition indicates the bit position of the least significant bit in an IPDU.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	--	
	LITTLE_ENDIAN	--	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00121 :**		
Name	IpduMIPduUnusedAreasDefault		
Description	IpduM module fills not used areas of an I-PDU with this bit-pattern If this attribute is omitted the IpduM module does not fill the I-PDU.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00158 :**		
Name	IpduMTxConfirmationPduId		
Description	Handle Id used by the PduR for confirmation (IpduM_TxConfirmation) and for TriggerTransmit (IpduM_TriggerTransmit). The existence of this parameter is essential for the PduR generation tool to actually find a symbolicNameValue for the OutgoingPdu.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00124 :**
Name	IpduMTxConfirmationTimeout
Description	This timeout (in seconds) defines the timeout period for monitoring the reception of the TxConfirmation.

	It is not used when an I-PDU is requested using the trigger transmit API.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. 3600		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00125 :**		
Name	IpduMTxTriggerMode		
Description	Selects whether to send the multiplexed I-PDU immediately or at some later date.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DYNAMIC_PART_TRIGGER	Writing the I-PDU representing the dynamic part does trigger a sending of the I-PDU.	
	NONE	Only the buffer in the IpduM are written but not send is triggered, used for IpduM I-PDUs which are requested by TriggerTransmit.	
	STATIC_OR_DYNAMIC_PART_TRIGGER	Writing the I-PDU representing the static or the dynamic part does trigger a sending of the I-PDU.	
	STATIC_PART_TRIGGER	Writing the I-PDU representing the static part does trigger a sending of the I-PDU.	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00157 :**		
Name	IpduMInitialDynamicPart		
Description	Reference to the dynamic part that shall be used to initialize this multiple-xed TX-I-PDU.		
Multiplicity	1		
Type	Reference to [IpduMTxDynamicPart]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00120 :**		
Name	IpduMOutgoingPduRef		
Description	Reference to the PDU defining the outgoing I-PDU. When the outgoing I-PDU is sent this is the I-PDU ID to give it. It is the IpduM I-PDU ID of the assembled I-PDU.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMSelectorField	1	Specifies the position of the selector field in the outgoing I-PDU.
IpduMTxDynamicPart	1..*	This (These) included container(s) must exist for each unique selector field value for this outgoing IpduM I-PDU.
IpduMTxDynamicSegment	1..*	The dynamic part of the multiplexed outgoing I-Pdu (referenced by IpduMOutgoingPduRef) can be separated into several segments. For each segment one IpduMTxDynamicSegment container shall be created that contains the location and the length of the segment. Please note that each configured segment will be copied out of the source I-Pdu that is referenced in the IpduMTxDynamicPart container and will be copied to the same location in the multiplexed outgoing I-Pdu. The segment layout for all dynamic Parts is always identical.
IpduMTxStaticPart	0..1	This included container configures the static part, if present.
IpduMTxStaticSegment	0..*	The static part of the multiplexed outgoing I-Pdu (referenced by IpduMOutgoingPduRef) can be separated into several segments. For each segment one IpduMTxStaticSegment container shall be created that contains the location and the length of the segment. Please note that each segment in the source I-Pdu that is referenced in the IpduMTxStaticPart container will be copied to the same location in the multiplexed outgoing I-Pdu.

10.2.8 IpduMTxDynamicPart

SWS Item	**ECUC_IpduM_00056 :**
Container Name	IpduMTxDynamicPart
Description	Configuration parameters for an instance of a TxRequest call into the IpduM. When a Tx Request with the IpduMTxDynamicHandleId is received by the IpduM, all segments (defined in the IpduMDynamicSegment container) are copied from the incoming I-PDU into the outgoing I-PDU buffer and then the send mode honored. This container is used by the dynamic part of a TxRequest configuration. Therefore, for each outgoing I-PDU there will be one instance of this container for the dynamic part. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00167 :**
Name	IpduMJitUpdate
Description	If configured to true fetch the data of this part Just-In-Time via the trigger-

	Transmit API of the PduR.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00163 :**		
Name	IpduMTxDynamicConfirmation		
Description	A transmit request can be confirmed by the lower layer. If this parameter is set to true a confirmation of the I-PDU in COM representing the dynamic part is generated.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00127 :**		
Name	IpduMTxDynamicHandleId		
Description	This defines an incoming handle id. When the handle of an incoming Tx Request matches this id, the configured dynamic segments are copied and the IpduMTxTriggerMode is honored.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_IpduM_00126 :**		
Name	IpduMTxDynamicPduRef		
Description	Reference to the Pdu representation in the ECU Configuration Description exchange file to be transmitted.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMSegment	0..*	This is a list of all segments to be copied from the incoming I-PDU to the outgoing I-PDU.

10.2.9 IpduMTxDynamicSegment

SWS Item	**ECUC_IpduM_00168 :**
Container Name	IpduMTxDynamicSegment
Description	The dynamic part of the multiplexed outgoing I-Pdu (referenced by IpduMOutgoingPduRef) can be separated into several segments. For each segment one IpduMTxDynamicSegment container shall be created that contains the location and the length of the segment. Please note that each configured segment will be copied out of the source I-Pdu that is referenced in the IpduMTxDynamicPart container and will be copied to the same location in the multiplexed outgoing I-Pdu. The segment layout for all dynamic Parts is always identical. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00114 :**		
Name	IpduMSegmentLength		
Description	Length of the segment in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2032		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00159 :**		
Name	IpduMSegmentPosition		
Description	Segments bit position in the multiplexed Pdu.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.10 IpduMTxStaticPart

SWS Item	**ECUC_IpduM_00082 :**
Container Name	IpduMTxStaticPart
Description	Configuration parameters for an instance of a Tx_Request call into the IpduM. When a Tx Request with the IpduMTxStaticHandleId is received by the IpduM, all segments (defined in the IpduMStaticSegment container) are copied from the incoming I-PDU into the outgoing I-PDU buffer and

	then the send mode honored. This container is used for the static part of a TxRequest configuration. Therefore, for each outgoing I-PDU there will be one instance of this container for the static part if it exists.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00167 :**		
Name	IpduMJitUpdate		
Description	If configured to true fetch the data of this part Just-In-Time via the trigger-Transmit API of the PduR.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00164 :**		
Name	IpduMTxStaticConfirmation		
Description	A transmit request can be confirmed by the lower layer. If this parameter is set to true a confirmation of the I-PDU in COM representing the static part is generated.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00129 :**		
Name	IpduMTxStaticHandleId		
Description	This defines an incoming handle id. When the handle of an incoming Tx Request matches this id, the configured static segments are copied and the IpduMTxTriggerMode is honored.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_IpduM_00128 :**		
Name	IpduMTxStaticPduRef		
Description	Reference to the Pdu representation in the ECU Configuration Description exchange file to be transmitted.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMSegment	0..*	This is a list of all segments to be copied from the incoming I-PDU to the outgoing I-PDU.

10.2.11 IpduMTxStaticSegment

SWS Item	**ECUC_IpduM_00171 :**
Container Name	IpduMTxStaticSegment
Description	The static part of the multiplexed outgoing I-Pdu (referenced by IpduMOut-goingPduRef) can be separated into several segments. For each segment one IpduMTxStaticSegment container shall be created that contains the location and the length of the segment. Please note that each segment in the source I-Pdu that is referenced in the IpduMTxStaticPart container will be copied to the same location in the multiplexed outgoing I-Pdu. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00114 :**		
Name	IpduMSegmentLength		
Description	Length of the segment in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2032		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00159 :**		
Name	IpduMSegmentPosition		
Description	Segments bit position in the multiplexed Pdu.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.12 IpduMRxPathway

SWS Item	**ECUC_IpduM_00071 :**
Container Name	IpduMRxPathway
Description	Contains the configuration parameters received I-PDUs by the IpduM module. **Attributes:** postBuildChangeable=true
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMRxIndication	1	configuration for RxIndication

10.2.13 IpduMRxIndication

SWS Item	**ECUC_IpduM_00047 :**
Container Name	IpduMRxIndication
Description	Contains the configuration for incoming RxIndication calls.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00162 :**		
Name	IpduMByteOrder		
Description	This parameter defines the ByteOrder for all segments (static and dynamic part) and for the selectorField within the MultiplexedPdu. The absolute position of a segment in the MultiplexedIPdu is determined by the definition of the ByteOrder parameter: If BIG_ENDIAN is specified, the SegmentPosition indicates the bit position of the most significant bit in an IPDU. If LITTLE_ENDIAN is specified, the SegmentPosition indicates the bit position of the least significant bit in an IPDU.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	--	
	LITTLE_ENDIAN	--	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00109 :**		
Name	IpduMRxHandleId		
Description	This is the I-PDU ID of the incoming I-PDU. If an incoming RxIndication's I-PDU ID matches this value then it is unpacked according to the specification in this container.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-

			BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_IpduM_00108 :**		
Name	IpduMRxIndicationPduRef		
Description	Reference to the received Pdu representation in the ECU Configuration Description exchange file.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMRxDynamicPart	0..*	Each of these containers contains the configuration for one value of the selector field for the incoming I-PDU's dynamic part.
IpduMRxDynamicSegment	0..*	The dynamic part of the multiplexed incoming I-Pdu (referenced by IpduMRxIndicationPduRef) can be separated into several segments. For each segment one IpduMRxDynamicSegment container shall be created that contains the location and the length of the segment. Please note that each configured segment will be copied into the destination I-Pdu that is referenced in the IpduMRxDyna-micPart container and will be copied from the same location in the multiplexed incoming I-Pdu. The segment layout for all dynamic Parts is always identical.
IpduMRxStaticPart	0..1	This contains the configuration for the incoming I-PDU's static part. If the incoming I-PDU has no static part then this is omit-ted.
IpduMRxStaticSegment	0..*	The static part of the multiplexed incoming I-Pdu (referenced by IpduMRxIndicationPduRef) can be separated into several seg-ments. For each segment one IpduMRxStaticSegment container shall be created that contains the location and the length of the seg-ment. Please note that each configured segment will be copied into the destination I-Pdu that is referenced in the IpduMRxSta-ticPart container and will be copied from the same location in the multiplexed incoming I-Pdu.
IpduMSelectorField	1	This contains the location of the selector field. At run-time, the selector field is used to select which dynamic part is unpacked.

10.2.14 IpduMRxDynamicPart

SWS Item	**ECUC_IpduM_00048 :**
Container Name	IpduMRxDynamicPart

Description	This container contains the configuration for the dynamic part of incoming RxIndication calls. When an incoming received I-PDU's selector field matches the IpduMRxSelectorValue, the new outgoing I-PDU for the dynamic part is constructed as defined by the segments (defined in the IpduMDynamicSegment container) and sent out with the I-PDU ID referenced by IpduMOutgoingDynamicPduRef. In case no dynamic part shall be extracted from this received I-PDU this container does not exist. This use-case can occur in case a MultiplexedIPdu is received by an ECU which is only interested in the static part of the MultiplexedIPdu. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00113 :**		
Name	IpduMRxSelectorValue		
Description	This is the selector value that this container refers to.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00112 :**		
Name	IpduMOutgoingDynamicPduRef		
Description	When the new I-PDU is sent out it is sent with this I-PDU ID. Reference to the sent PDU representation in the ECU Configuration Description exchange file.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMSegment	0..*	The DynamicPart can be separated in multiple segments within the multiplexed PDU.

10.2.15 IpduMRxDynamicSegment

SWS Item	**ECUC_IpduM_00170 :**
Container Name	IpduMRxDynamicSegment
Description	The dynamic part of the multiplexed incoming I-Pdu (referenced by IpduMRxIndicationPduRef) can be separated into several segments. For each segment one IpduMRxDynamicSegment container shall be created

	that contains the location and the length of the segment. Please note that each configured segment will be copied into the destination I-Pdu that is referenced in the IpduMRxDynamicPart container and will be copied from the same location in the multiplexed incoming I-Pdu. The segment layout for all dynamic Parts is always identical. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00114 :**		
Name	IpduMSegmentLength		
Description	Length of the segment in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2032		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00159 :**		
Name	IpduMSegmentPosition		
Description	Segments bit position in the multiplexed Pdu.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.16 IpduMRxStaticPart

SWS Item	**ECUC_IpduM_00049 :**
Container Name	IpduMRxStaticPart
Description	This container contains the configuration for the static part of incoming RxIndication calls. On reception, the new outgoing I-PDU for the static part is constructed as defined by the segments (defined in the IpduMStaticSegment container) and sent out with the I-PDU ID referenced by IpduMOutgoingStaticPduRef.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00115 :**
Name	IpduMOutgoingStaticPduRef
Description	When the new I-PDU is sent out it is sent with this I-PDU ID. Reference to the sent Pdu representation in the ECU Configuration Description exchan-

	ge file.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
IpduMSegment	0..*	The StaticPart can be separated in multiple segments within the multiplexed PDU.

10.2.17 IpduMRxStaticSegment

SWS Item	**ECUC_IpduM_00169 :**
Container Name	IpduMRxStaticSegment
Description	The static part of the multiplexed incoming I-Pdu (referenced by IpduMRxIndicationPduRef) can be separated into several segments. For each segment one IpduMRxStaticSegment container shall be created that contains the location and the length of the segment. Please note that each configured segment will be copied into the destination I-Pdu that is referenced in the IpduMRxStaticPart container and will be copied from the same location in the multiplexed incoming I-Pdu. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00114 :**		
Name	IpduMSegmentLength		
Description	Length of the segment in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2032		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00159 :**		
Name	IpduMSegmentPosition		
Description	Segments bit position in the multiplexed Pdu.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local

No Included Containers

10.2.18 IpduMSegment

SWS Item	**ECUC_IpduM_00053 :**
Container Name	IpduMSegment
Description	Please note that this container is deprecated and will be removed in the future. Old description: This contains the location and the length of a segment. A segment must fit inside the I-PDU. The segment in the source I-PDU that is located at the IpduMSegmentPosition is copied to the same position in the destination I-PDU. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_IpduM_00114 :**		
Name	IpduMSegmentLength		
Description	Length of the segment in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2032		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00159 :**		
Name	IpduMSegmentPosition		
Description	Segments bit position in the multiplexed Pdu.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.19 IpduMSelectorField

SWS Item	**ECUC_IpduM_00054 :**

Container Name	IpduMSelectorField
Description	This contains the location and the length of the selector field.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00160 :**		
Name	IpduMSelectorFieldLength		
Description	Length of the selector field in bits.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 8		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_IpduM_00161 :**		
Name	IpduMSelectorFieldPosition		
Description	Selector field bit position in the multiplexed Pdu.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

For details refer to the Chapter 10.3 Published Information in SWS_BSWGeneral.

10.3.1 IpduMPublishedInformation

SWS Item	**ECUC_IpduM_00141 :**
Container Name	IpduMPublishedInformation
Description	Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, since they are published information.
Configuration Parameters	

SWS Item	**ECUC_IpduM_00142 :**		
Name	IpduMRxDirectComInvocation		
Description	If set to TRUE the COM invocation optimization as defined in IPDUM140 is implemented.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Published Information	X	All Variants
Scope / Dependency	scope: local		

No Included Containers

10.4 Configuration Rules

10.4.1 Selector Field

[SWS_IpduM_00155] ⌈The selector fields shall not cross any byte-boundary within the I-PDU.⌋ (SRS_IpduM_02802)

Restricting the selector field to be within one byte helps avoiding endianness related problems regarding the selector field.

[SWS_IpduM_00011] ⌈The number of values used of the selector field, i.e. values used to distinguish between different I-PDU layouts, does not have to be the whole range of possible values.⌋ (SRS_IpduM_02803)

Example: The size of a selector field with 3 bits leads to 2^3 possible selector field values; it shall be allowed to use only an arbitrary subset of these values. The used subset needs no to be contiguous.

10.4.2 Byte Order

The byte order of all segments and the selector field of a multiplexed I-PDU is restricted to be the same, see ECUC_IpduM_00162. Any necessary byte order conversion shall be handled within the COM module. The multiplexed I-PDUs in COM and IpduM have to be configured consistently to have the same endianness.

[SWS_IpduM_00166] ⌈The endianness of signals of the de-multiplexed I-PDUs configured in COM must match the endianness of the corresponding multiplexed I-PDU in IpduM as configured per IpduMByteOrder (ECUC_IpduM_00162). ⌋ ()

The above configuration rule also restricts all COM signals of a multiplexed attribute to have the same endianness.

11 Not applicable requirements

[SWS_IpduM_00999]「These requirements are not applicable to this specification.」(SRS_BSW_00171, SRS_BSW_00375, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00427, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00386, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00314, SRS_BSW_00377)