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

Document Version	1.5.1
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	1.5.1	AUTOSAR Release Management	• Formal and editorial corrections
17.05.2012	1.5.0	AUTOSAR Administration	• Enabled use-case to receive only the static part of a multiplexed I-PDU • Minor bug fixes and editorial changes
07.04.2011	1.4.0	AUTOSAR Administration	• Harmonized IpduM-prefix • Added configurable JIT-update • Minor bug fixes and editorial changes • Legal disclaimer revised
16.09.2010	1.3.0	AUTOSAR Administration	• Added a pre-compile configuration variant • Added IPDUM162 in configuration container IpduMTxRequest and IpduMRxIndication • Updated IPDUM032, IPDUM060, IPDUM040, IPDUM043, IPDUM060 • Added IPDUM163 • 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

Document Change History

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

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

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 part 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 [6]
Static part	see [6]
Selector field	see [6]
Signal	see [7]
Signal group	see [7]
Sub part	The static or dynamic part may consist of more than one element. These sub-elements are called sub-parts; see also IPDUM006 and Figure 2.
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	IpduM I-PDU with one specific layout and content

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_General.pdf

[4] Specification of ECU Configuration
AUTOSAR_ECU_Configuration.pdf

[5] Specification of BSW Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

[6] AUTOSAR Requirements on I-PDU Multiplexer
AUTOSAR_SRS_IPDUM.pdf

[7] AUTOSAR Specification of Communication
AUTOSAR_SWS_COM.pdf

[8] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

None

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 [7] 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 of the position of the IpduM module in the layered architecture, it has only interfaces to the PDU-Router. See also 7.2.
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.

5.1 AUTOSAR OS

IPDUM107: The IpduM shall not directly access the AUTOSAR OS.

5.2 BSW Scheduler

IpduM shall use the BSW-Scheduler to schedule its main functions, see also [5].

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

5.4 File structure

5.4.1 Code file structure

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

IPDUM095: The module IpduM shall provide a file IpduM_Lcfg.c containing the link-time configurable parameters.

IPDUM096: The module IpduM shall provide a file IpduM_PBcfg.c containing the post-build time configurable parameters.

5.4.2 Header file structure

IPDUM002: The IpduM module shall comply with the following include-file structure:

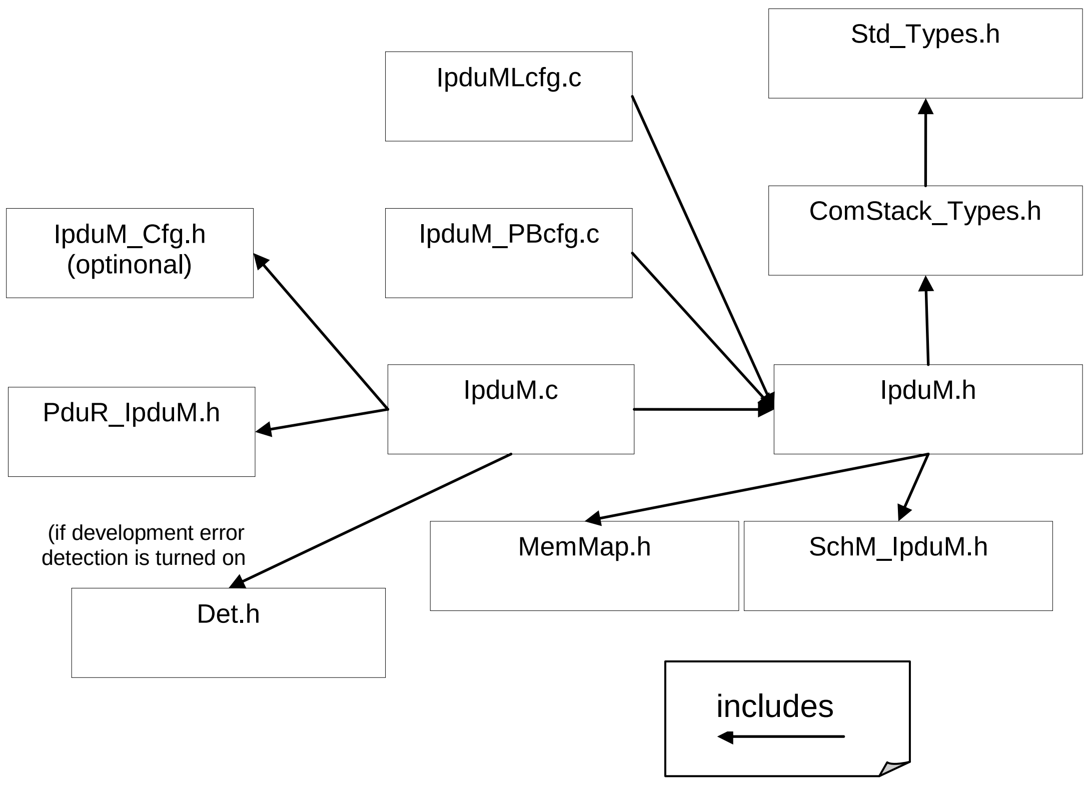

Figure 1 Header File Structure

5.4.3 Design Rules

IPDUM073: The code of the IpduM module, as long as it is written in C, shall conform to the HIS subset of the MISRA C Standard.

IPDUM074: The code of the IpduM module shall avoid direct use of compiler and platform specific keywords.

IPDUM075: The code of the IpduM module shall indicate all global data with read-only purposes by explicitly assigning the const keyword.

IPDUM076: The IpduM module can use macros instead of functions where source code is used and runtime is critical.

IPDUM077: The IpduM module shall not define global data in the header files. If global variables are used, the definition shall take place in the C file.

IPDUM078: The source code of the IpduM module shall not be processor and compiler dependent.

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software [3]

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	Chapter 10.2.2, IPDUM032
[BSW00404] Reference to post build time configuration	Chapter 10.2
[BSW00405] Reference to multiple configuration sets	IPDUM032
[BSW00345] Pre-compile-time configuration	Chapter 10.2.2, IPDUM059, IPDUM047, IPDUM048, IPDUM049, IPDUM050, IPDUM051, IPDUM052, IPDUM053, IPDUM056
[BSW159] Tool-based configuration	not scope of this specification Refers to Configuration WP.
[BSW167] Static configuration checking	not scope of this specification Refers to Configuration WP.
[BSW171] Configurability of optional functionality	not applicable (there is no optional functionality)
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	not scope of this specification Refers to Configuration WP.
[BSW00380] Separate C-Files for configuration parameters	IPDUM095, IPDUM096 implementation specific
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Chapter 5.4 implementation specific
[BSW00381] Separate configuration header file for pre-compile time parameters	Chapter 5.4 implementation specific
[BSW00412] Separate H-File for configuration parameters	Chapter 5.4 implementation specific
[BSW00383] List dependencies of configuration files	not scope of this specification
[BSW00384] List dependencies to other modules	Chapter 5, IPDUM104, IPDUM105
[BSW00387] Specify the configuration class of callback function	Chapter 8.5
[BSW00388] Introduce containers	Chapter 10.2,IPDUM070, IPDUM071, IPDUM082, IPDUM130
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2
[BSW00391] Parameter shall have unique names	Chapter 10.2
[BSW00392] Parameters shall have a type	Chapter 10.2
[BSW00393] Parameters shall have a range	Chapter 10.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2
[BSW00395] List the required parameters (per parameter)	All parameter in Chapter 10.2 are required.
[BSW00396] Configuration classes	Chapter 10.2

[BSW00397] Pre-compile-time parameters	Chapter 10.2
[BSW00398] Link-time parameters	Chapter 10.2
[BSW00399] Loadable Post-build time parameters	Chapter 10.2
[BSW00400] Selectable Post-build time parameters	Chapter 10.2
[BSW00438] Post Build Configuration Data Structure	Chapter 10.2.1
[BSW00402] Published information	Chapter 10.3
[BSW00375] Notification of wake-up reason	not applicable (this layer can not perform a wake-up)
[BSW101] Initialization interface	IPDUM032, IPDUM033, IPDUM034, IPDUM035, IPDUM064, IPDUM065, IPDUM092
[BSW00416] Sequence of Initialization	not scope of this specification refere to Mode Management Specification.
[BSW00406] Check module initialization	IPDUM083, IPDUM084
[BSW00437] NoInit—Area in RAM	not applicable (not needed)
[BSW168] Diagnostic interface	not applicable (not diagnostic interface included)
[BSW00407] Function to read out published parameters	IPDUM037
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	not applicable (this module has no connection to the RTE)
[BSW00424] BSW main processing function task allocation	not scope of this specification Implementation specific
[BSW00425] Trigger conditions for schedulable objects	IPDUM103, IPDUM131
[BSW00426] Exclusive areas in BSW modules	not scope of this specification Implementation specific
[BSW00427] ISR description for BSW modules	not applicable (module does not provide ISRs)
[BSW00428] Execution order dependencies of main processing functions	Chapter 8.6
[BSW00429] Restricted BSW OS functionality access	IPDUM107
[BSW00431] The BSW Scheduler module implements task bodies	not applicable (requirement for the scheduler)
[BSW00432] 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)
[BSW00433] 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)
[BSW00336] Shutdown interface	not applicable (not needed)
[BSW00337] Classification of errors	IPDUM026, IPDUM106
[BSW00338] Detection and Reporting of development errors	IPDUM027, IPDUM028, IPDUM059, IPDUM132

[BSW00369] Do not return development error codes via API	IPDUM032, IPDUM037, IPDUM040, IPDUM043, IPDUM044, IPDUM060
[BSW00339] Reporting of production relevant errors and exceptions	IPDUM029, IPDUM030
[BSW00422] Pre—de—bouncing of production relevant error status	not applicable (not scope of this specification)
[BSW00417] Reporting of Error Events by Non-Basic Software	not applicable (this module is part of the basic software)
[BSW00323] API parameter checking	IPDUM028
[BSW004] Version check	IPDUM057, IPDUM038, IPDUM039, IPDUM059, IPDUM134
[BSW00409] Header files for production code error IDs	Figure 1
[BSW00385] List possible error notifications	IPDUM026
[BSW00386] Configuration for detecting an error	not applicable (implementation specific)
[BSW161] Microcontroller abstraction	IPDUM074, IPDUM078
[BSW162] ECU layout abstraction	not applicable (not scope of this specification)
[BSW005] No hard coded horizontal interfaces within MCAL	not applicable (not scope of this specification)
[BSW00415] User dependent include files	IPDUM002
[BSW164] Implementation of interrupt service routines	not applicable (module does not provide ISRs)
[BSW00325] Runtime of interrupt service routines	not applicable (module does not provide ISRs)
[BSW00326] Transition from ISRs to OS tasks	not applicable (module does not provide ISRs)
[BSW00342] Usage of source code and object code	Chapter 10.2
[BSW00343] Specification and configuration of time	Chapter 10.2
[BSW160] Human-readable configuration data	Chapter 10.2
[BSW007] HIS MISRA C	IPDUM073
[BSW00300] Module naming convention	Figure 1
[BSW00413] Accessing instances of BSW modules	not scope of this specification implementation specific
[BSW00347] Naming separation of different instances of BSW drivers	not scope of this specification implementation specific
[BSW00305] Self-defined data types naming convention	Chapter 8.3.1
[BSW00307] Global variables naming convention	not scope of this specification implementation specific
[BSW00310] API naming convention	Chapter 8.4 and 8.5
[BSW00373] Main processing function naming convention	Chapter 8.6
[BSW00327] Error values naming convention	IPDUM026
[BSW00335]	not scope of this specification

Status values naming convention	implementation specific
[BSW00350] Development error detection keyword	IPDUM027
[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00410] Compiler switches shall have defined values	not scope of this specification implementation specific
[BSW00411] Get version info keyword	IPDUM039
[BSW00346] Basic set of module files	Figure 1
[BSW158] Separation of configuration from implementation	Figure 1
[BSW00314] Separation of interrupt frames and service routines	not applicable (module does not provide ISRs)
[BSW00370] Separation of callback interface from API	Chapter 8.5
[BSW00435] Module Header File Structure for the Basic Software Scheduler	Figure 1.
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Figure 1.
[BSW00348] Standard type header	Figure 1
[BSW00353] Platform specific type header	not scope of this specification implementation specific
[BSW00361] Compiler specific language extension header	not scope of this specification implementation specific
[BSW00301] Limit imported information	not scope of this specification implementation specific
[BSW00302] Limit exported information	not scope of this specification implementation specific
[BSW00328] Avoid duplication of code	not scope of this specification implementation specific
[BSW00312] Shared code shall be reentrant	not scope of this specification implementation specific
[BSW006] Platform independency	not scope of this specification implementation specific
[BSW00357] Standard API return type	Chapter 8, IPDUM102
[BSW00377] Module specific API return types	not applicable (no specific return types)
[BSW00304] AUTOSAR integer data types	Figure 1 implementation specific
[BSW00355] Do not redefine AUTOSAR integer data types	Chapter 8.3 implementation specific
[BSW00378] AUTOSAR boolean type	not scope of this specification implementation specific
[BSW00306] Avoid direct use of compiler and platform specific keywords	not scope of this specification implementation specific
[BSW00308] Definition of global data	not scope of this specification implementation specific
[BSW00309] Global data with read-only constraint	IPDUM075, IPDUM077
[BSW00371] Do not pass function pointers via API	Chapter 8.4 and 8.5
[BSW00358] Return type of init functions	Chapter 8.4.1

[BSW00414] Parameter of init function	Chapter 8.4.1
[BSW00376] Return type and parameters of main processing functions	Chapter 8.6
[BSW00359] Return type of callback functions	Chapter 8.5
[BSW00360] Parameters of callback functions	Chapter 8.5
[BSW00329] Avoidance of generic interfaces	Chapter 8
[BSW00330] Usage of macros / inline functions instead of functions	IPDUM076, IPDUM085
[BSW00331] Separation of error and status values	Chapter 8
[BSW009] Module User Documentation	not scope of this specification implementation specific
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[BSW172] Compatibility and documentation of scheduling strategy	not scope of this specification implementation specific
[BSW010] Memory resource documentation	not scope of this specification implementation specific
[BSW00333] Documentation of callback function context	not scope of this specification implementation specific
[BSW00374] Module vendor identification	Chapter 10.3
[BSW00379] Module identification	Chapter 10.3
[BSW003] Version identification	IPDUM037, IPDUM057, IPDUM059
[BSW00318] Format of module version numbers	Chapter 10.3
[BSW00321] Enumeration of module version numbers	not scope of this specification implementation specific
[BSW00341] Microcontroller compatibility documentation	not scope of this specification implementation specific
[BSW00334] Provision of XML file	not scope of this specification Refers to Configuration WP

Document: AUTOSAR requirements on Basic Software cluster IPDUM [6]

Requirement	***Satisfied by***
[BSW02800] Exactly one selector field per PDU	IPDUM004, IPDUM007
[BSW02801] Size of the selector field	IPDUM009, IPDUM052
[BSW02802] Position of the selector field	IPDUM005
[BSW02815] Compile Time configuration of the selector field	IPDUM052
[BSW02803] Unused values of the selector field	IPDUM011
[BSW02804] Support for static and dynamic parts of the PDU	IPDUM006, IPDUM008

Requirement	*Satisfied by*
[BSW02808] Support of multiplexed PDUs with a static part of length “zero”	IPDUM004, IPDUM133
[BSW02809] Initialization of multiplexed PDUs	IPDUM013, IPDUM069, IPDUM068, IPDUM067, IPDUM098, IPDUM099
[BSW02806] Semantic of the multiplexer	IPDUM010
[BSW02810] Routing of multiplexed PDUs on sender side	IPDUM063, IPDUM089, IPDUM090, IPDUM091, IPDUM112,
[BSW02816] Combining of multiplexed PDUs on sender side	IPDUM015, IPDUM017, IPDUM114, IPDUM120, IPDUM121, IPDUM122, IPDUM123, IPDUM124, IPDUM125, IPDUM126, IPDUM127, IPDUM128, IPDUM129, IPDUM167, IPDUM168, IPDUM169
[BSW02811] Triggering condition on sender side	IPDUM021, IPDUM052
[BSW02812] Routing of multiplexed PDUs on receiver side	IPDUM041, IPDUM042, IPDUM086, IPDUM108, IPDUM109
[BSW02817] De-multiplexing PDUs on receiver side	IPDUM040, IPDUM113, IPDUM114, IPDUM115, IPDUM170
[BSW02813] Routing of Send Confirmations	IPDUM022, IPDUM050, IPDUM072, IPDUM101, IPDUM117
[BSW02818] Confirmation replication of multiplexed PDUs	IPDUM022, IPDUM050, IPDUM051, IPDUM118, IPDUM119
[BSW02814] Correct confirmation handling of multiplexed PDUs	IPDUM023, IPDUM024, IPDUM019, IPDUM020, IPDUM087,IPDUM088
[BSW02807] No Runtime Overhead for systems without PDU multiplexing	IPDUM097
[BSW02819] No queuing of transmission requests on sender side	IPDUM020, IPDUM023

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 [2].

IPDUM004: 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 2.

Note: 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.

IPDUM005: 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 2, for configuration see Chapter 10.2.2.

IPDUM006: 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 *sub parts*.

IPDUM007: There shall be only one selector field within one multiplexed I-PDU.

IPDUM008: The value of the selector field shall define how the content of the dynamic part of the I-PDU shall be interpreted.

IPDUM009: The selector field of one I-PDU shall have a configurable size between one and eight contiguous bits.

IPDUM010: The position of the selector field within the I-PDU shall be defined by configuration.

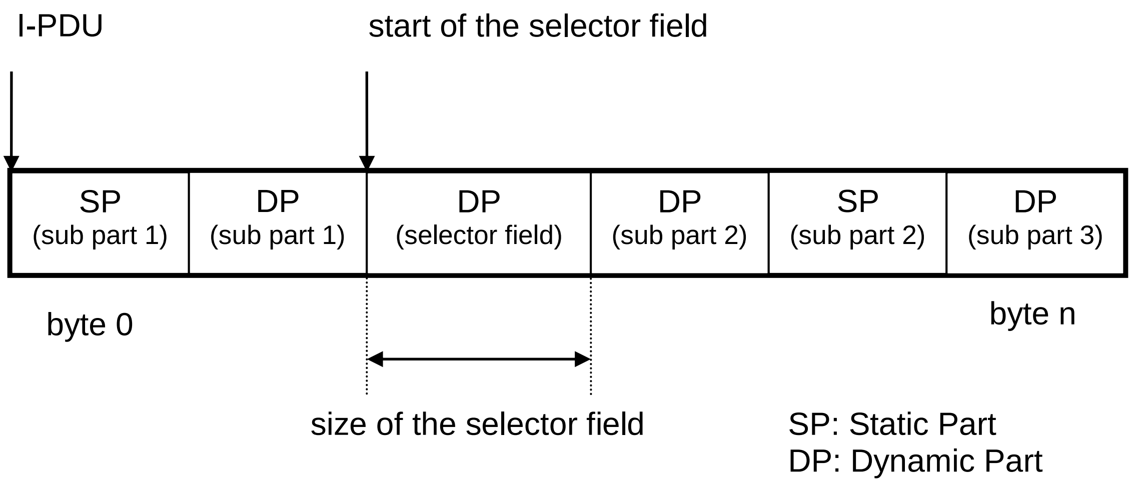

Figure 2 Possible layout of a multiplexed I-PDU

IPDUM011: 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.

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 a part of these values.

Note: 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.

7.2 Overview

The IpduM is arranged next to the PDU-Router in the layered architecture of AUTOSAR; see [2] and Figure 3.

IPDUM097: 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.

IPDUM013: The configuration of COM shall be such that each part of a multiplexed I-PDU, the static part and the different dynamic parts, are configured as different I-PDUs in COM.

Note: There is one COM I-PDU for the static part and one COM I-PDU for each layout of the dynamic part of one IpduM I-PDU, so the IpduM always combines only two I-PDUs of COM.

IPDUM098: The IpduM module shall not set the selector field.

IPDUM099: The configuration of COM shall be such that the selector field is part of the COM I-PDU for the dynamic part.

Note: This could be realized by defining a signal for the selector field in each instance of the dynamic part. This signal is initialized with the default value by the configuration of COM but never written during runtime.

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

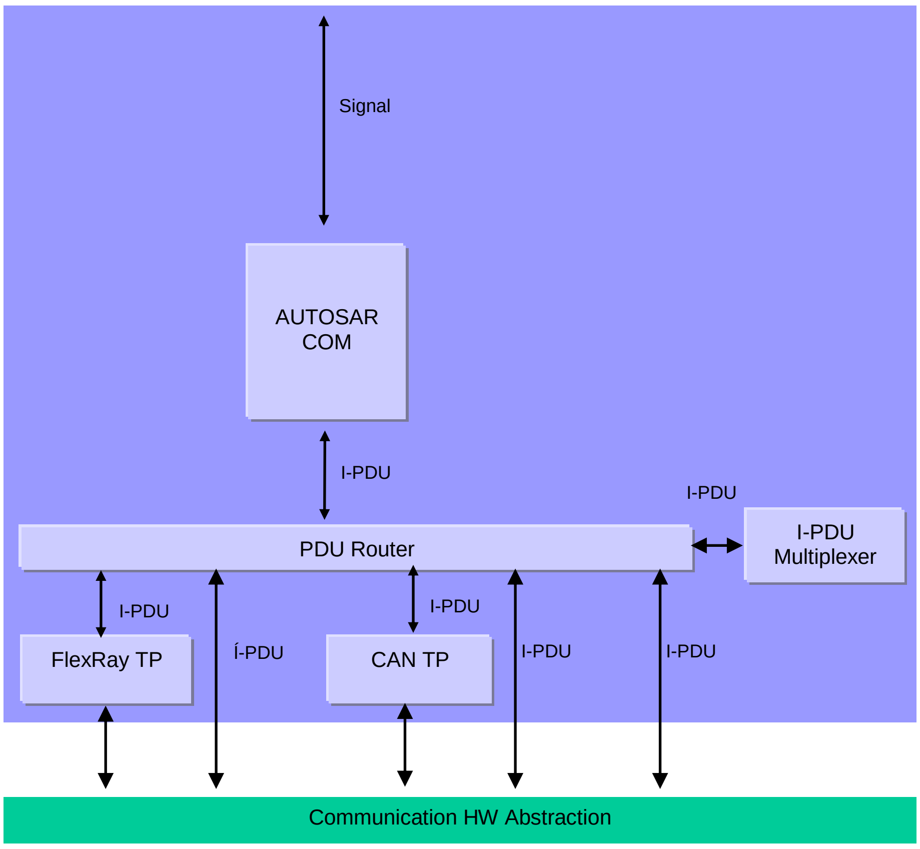

Figure 3 I-PDU Multiplexer in the Autosar Architecture

It should be allowed to optimize the receive and TxConfirmation 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.

7.3 Initialization

The IpduM module provides an initialization function IpduM_Init defined in IPDUM032. This function initializes all internal global variables and the buffers of the IpduM I-PDUs. For more details, see Chapter 8.3.1.

IPDUM092: The environment of the IpduM shall call IpduM_Init before calling any other function of the IPDUM module.

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 in case it is transmitted before it has been fully populated with data by COM. The initialization data for this buffer is arrived at as follows using configuration data from the IpduM_Tx_Request container.

1) **IPDUM067**: The buffer is first filled with the pattern defined in the configuration parameter IpduMIPduUnsuedAerasDefault.
2) **IPDUM068**: The initial selector field IpduMInitialSelectorValue is used to determine from COM’s configuration the initial value of the dynamic part. The initial value of the static part is determined by COM’s initial value for the incoming I-PDU.
3) **IPDUM069**: Finally, the selector field, indicated by the IpduMBitField container is filled with the value in the configuration parameter IpduMInitialSelectorValue.

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.

IPDUM063: The configuration of the PDU-Router module (e.g. look-up tables) shall 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.

IPDUM015: The IpduM module shall merge the two I-PDUs (the static part and the last received dynamic part) into one single I-PDU with a new unique I-PDU ID, which is sent out to the PDU-Router module.

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

Note: 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 IPDUM043.

IPDUM017: The function IpduM_Transmit (called with a COM I-PDU) shall assemble the related IpduM I-PDU, using the related static and dynamic part, and transmit it according to the trigger conditions.

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.

IPDUM019: The configuration of the IpduM shall contain a dedicated timeout for each IpduM I-PDU within the IpduM module in the configuration parameter IpduMTxConfirmationTimeout.

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.

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

IPDUM020: 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 not allow a new transmission request from the upper layer with a COM I-PDU that belongs to the same IpduM I-PDUs. In that case, the function IpduM_Transmit shall return with E_NOT_OK.

Note: It maybe 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 [7].

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.

IPDUM021: The IpduM module shall be configurable to send a transmission request for the new multiplexed I-PDU to the PDU-Router because of

- 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.

For configuration, see IPDUM052.

Note: By this mechanism, it is possible to control the transmission mode of the new assembled I-PDU by the transmission modes of the single I-PDUs sent by COM, see also [7].

Note: By this realization, it is not possible to 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 stand-alone I-PDUs, which are not connected together.

Note: 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.

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.

IPDUM168: 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.

IPDUM169: 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.

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 module look-up tables.

IPDUM022: 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.

Note: Depending on the configuration there are zero, one or two confirmations given to COM for one send request.

IPDUM023: If the TxConfirmation is not received within the configured timeout IpduMTxConfirmationTimeout the IpduM shall allow new transmission requests for this specific I-PDU after timeout is elapsed.

IPDUM024: The IpduM shall discard unexpected TxConfirmations silently. This may happen if a previously requested transmit has been timed out, but is confirmed now.

Note: There need not to be 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 Hardware Abstraction Layer (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 I-PDU would have to be routed to the IpduM as well.

7.6 Error classification

IPDUM026: 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]*

API service called with wrong parameter	Development	IPDUM_E_PARAM	10
API service used without module initialization	Development	IPDUM_E_UNINIT	20

IPDUM106: Development error values are of type uint8.

7.7 Error detection

IPDUM027: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch IpduMDevErrorDetect (see Chapter 10) shall activate or deactivate the detection of all development errors.

IPDUM028: If the IpduMDevErrorDetect switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in Chapter 7.6 and Chapter 8.

IPDUM029: The detection of production code errors cannot be switched off.

Note: Actually, there are no production errors defined for the IpduM.

7.8 Error notification

IPDUM030: The IpduM module shall report detected development errors to the error hook of the Development Error Tracer (DET) if the pre-processor switch IpduMDev-ErrorDetect is set, see Chapter 10.

8 API specification

8.1 Imported types

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

IPDUM102:

Module	*Imported Type*
ComStack_Types	PduIdType
	PduInfoType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

8.2.1 IpduM_ConfigType

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.

8.3 Function definitions

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

8.3.1 IpduM_Init

IPDUM032:

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.	

IPDUM033: The function IpduM_Init shall initialize all module-related global variables.

IPDUM034: The function IpduM_Init shall initialize all I-PDUs with the default values.

IPDUM035: The function IpduM_Init shall initialize the default value of the selector field with a configurable value.

IPDUM064: The function IpduM_Init shall initialize the states of the timeout monitors.

IPDUM065: The function IpduM_Init shall initialize the state of the TxConfirmation IDs.

IPDUM083: In case, the configuration parameter IpduMDevErrorDetect equals TRUE: if the parameter config does not correspond to a valid configuration, the function IpduM_Init shall raise the development error `IPDUM_E_PARAM`.

IPDUM084: The behavior of the IpduM is unspecified until a correct call to IpduM_Init is made.

8.3.2 IpduM_GetVersionInfo

IPDUM037:

Service name:	IpduM_GetVersionInfo	
Syntax:	void IpduM_GetVersionInfo(Std_VersionInfoType* versioninfo)	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.	

IPDUM038: The function IpduM_GetVersionInfo shall return the version information of this module. The version information includes:
- Module ID
- Vendor ID
- Vendor specific version numbers (BSW00407).

IPDUM039: The function IpduM_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: IpduMVersionInfoApi

IPDUM085: If source code for caller and callee of the function IpduM_GetVersionInfo are available, the module IpduM should realize this function as a macro, defined in the module's header file.

8.3.3 IpduM_Transmit

IPDUM043:

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 mutliplexed) - 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.	

For a detailed description read Chapter 7.4.1.

8.4 Call-back notifications

8.4.1 IpduM_RxIndication

IPDUM040:

Service name:	IpduM_RxIndication	
Syntax:	void IpduM_RxIndication(PduIdType PdumRxPduId, const PduInfoType* PduInfoPtr)	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.	
Parameters (in):	PdumRxPduId	ID of I-PDU that has been received.
	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:	Service is called when a multiplexed SDU has to be de-multiplexed.	

IPDUM041: 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.

IPDUM042: 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.

IPDUM170: Within the function IpduM_RxIndication, the IpduM module shall check the received data length (PduInfoPtr->SduLength) and process only completely received static or dynamic parts.

Note: The selector field is part of the dynamic part. Therefore, a dynamic part can only be received, if the selector field is also received.

IPDUM086: The function IpduM_RxIndication shall be callable in interrupt context, e.g. from receive interrupt.

8.4.2 IpduM_TxConfirmation

IPDUM044:

Service name:	IpduM_TxConfirmation	
Syntax:	void IpduM_TxConfirmation(PduIdType PdumTxPduId)	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.	
Parameters (in):	PdumTxPduId	ID of multiplexed I-PDU that has been transmitted. Range: 0..(maximum number of I-PDU IDs which are multiplexed) - 1
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the lower layer after the I-PDU has been transmitted on the network.	

IPDUM088: 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.

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

IPDUM087: The function IpduM_TxConfirmation shall be callable in interrupt context, e.g. from transmit interrupt.

8.4.3 IpduM_TriggerTransmit

IPDUM060:

Service name:	IpduM_TriggerTransmit
Syntax:	Std_ReturnType IpduM_TriggerTransmit(PduIdType PdumTxPduId,

	PduInfoType* PduInfoPtr)	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same PDU-ID. Reentrant for different PDU-ID.	
Parameters (in):	PdumTxPduId	ID of IpduM I-PDU that is requested to be transmitted by IpduM.
Parameters (inout):	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. SduLength has not been set.
Description:	Service is called by the lower layer when an IpduM I-PDU shall be transmitted.	

IPDUM090: The function IpduM_TriggerTransmit shall copy the contents of its I-PDU transmit buffer to the I-PDU buffer given by SduPtr.

IPDUM091: The IpduM shall take care about the data consistency during providing the data.

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 initiiated 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).

IPDUM089: The function IpduM_TriggerTransmit shall be callable in interrupt context.

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.

IPDUM103:

Service name:	IpduM_MainFunction
Syntax:	void IpduM_MainFunction( )
Service ID[hex]:	0x10
Timing:	FIXED_CYCLIC_WITH_PRECONDITION
Description:	Performs the processes of the activities that are not directly initiated by the calls from PDU-R.

IPDUM101: 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.

IPDUM072: The configuration of the BSWM Scheduler shall be such that the cycle time is equal to the smallest IpduMTxConfirmationTimeout.

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.

IPDUM104:

API function	*Description*
PduR_IpduMRxIndication	Rx indicator for the IpduM
PduR_IpduMTxConfirmation	Tx confirmation for the IpduM

8.6.2 Optional Interfaces

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

IPDUM105:

API function	*Description*
Det_ReportError	Service to report development errors.
PduR_IpduMTransmit	Requests a transmission for the IpduM
PduR_IpduMTriggerTransmit	IpduM requests the buffer of the SDU for transmission from the PduR.

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.

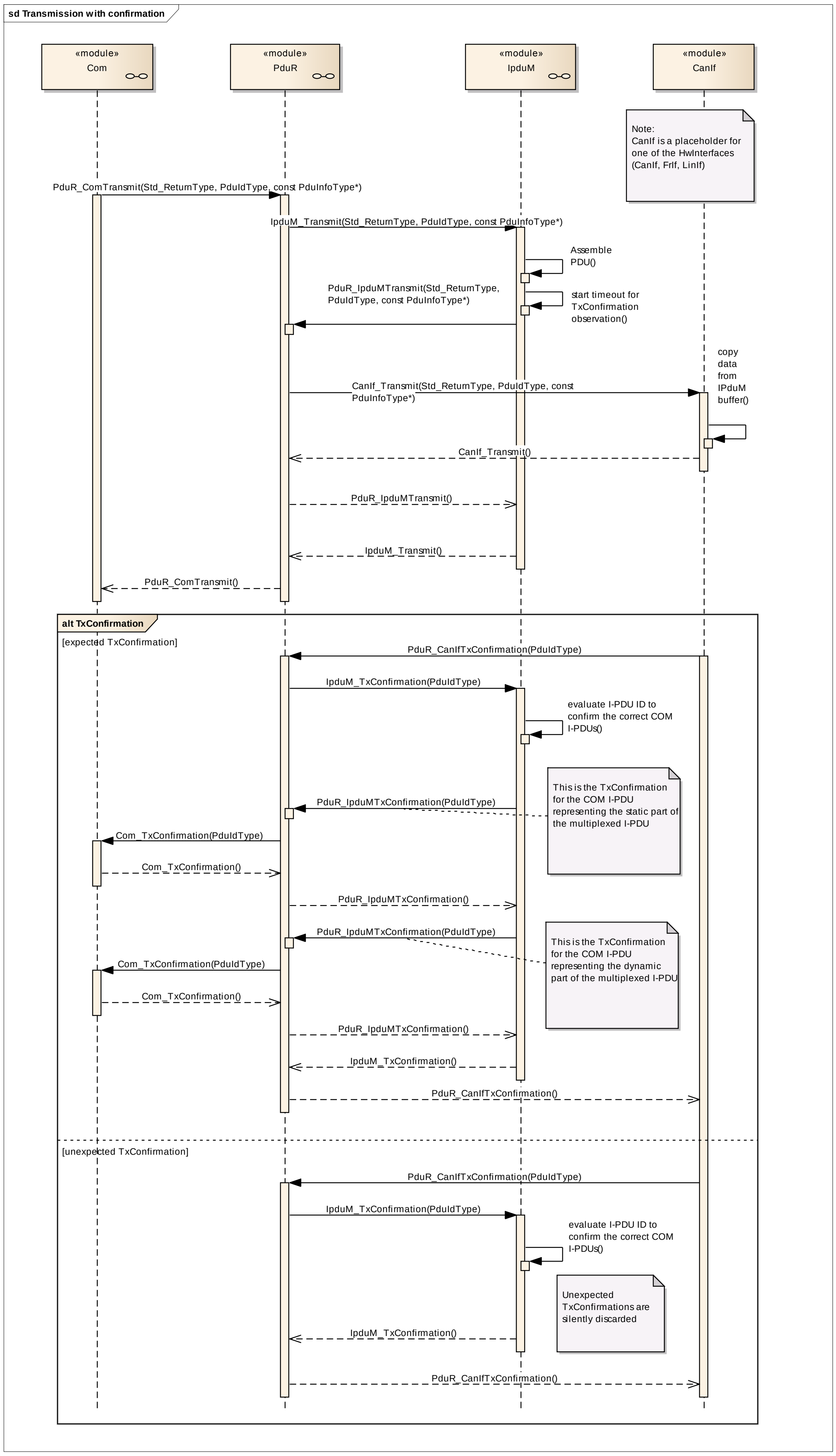
sd Transmission with confirmation
«module» Com
«module» PduR
«module» IpduM
«module» CanIf
Note: CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Assemble PDU()
PduR_IpduMTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
start timeout for TxConfirmation observation()
copy data from IPduM buffer()
CanIf_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
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 IPDUM052.

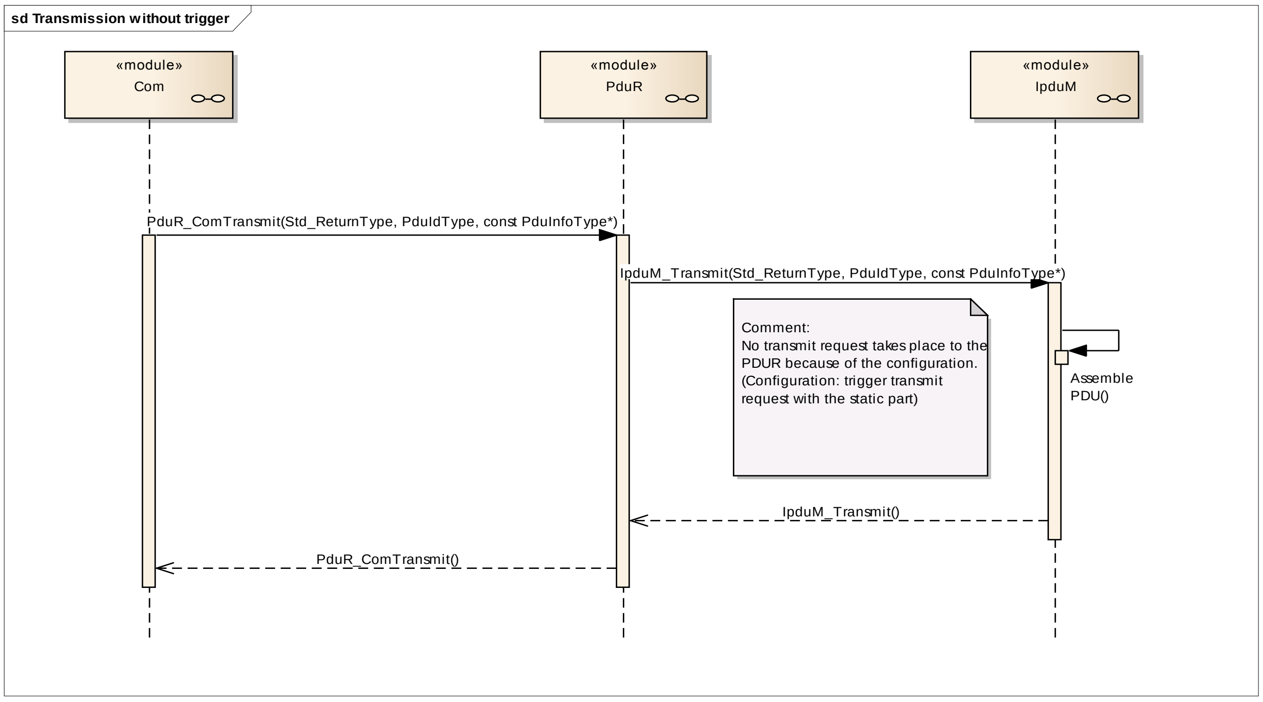
sd Transmission without trigger
«module» Com
«module» PduR
«module» IpduM
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Comment:
No transmit request takes place to the PDUR because of the configuration. (Configuration: trigger 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.

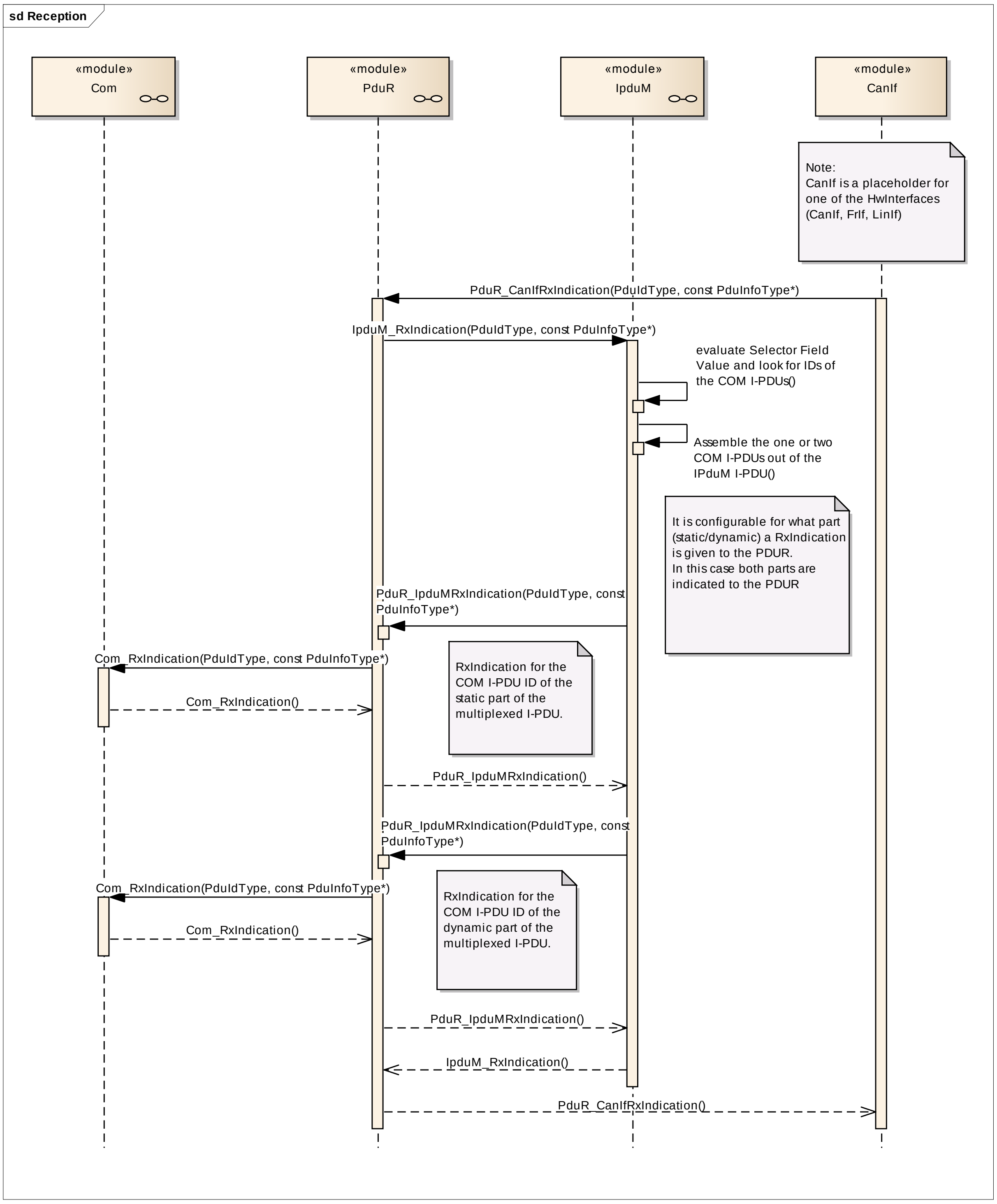
sd Reception
«module» Com
«module» PduR
«module» IpduM
«module» CanIf
Note:
CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
PduR_CanIfRxIndication(PduIdType, const PduInfoType*)
IpduM_RxIndication(PduIdType, const PduInfoType*)
evaluate Selector Field Value and look for IDs of 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 PduInfoType*)
Com_RxIndication(PduIdType, const 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 PduInfoType*)
Com_RxIndication(PduIdType, const 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.

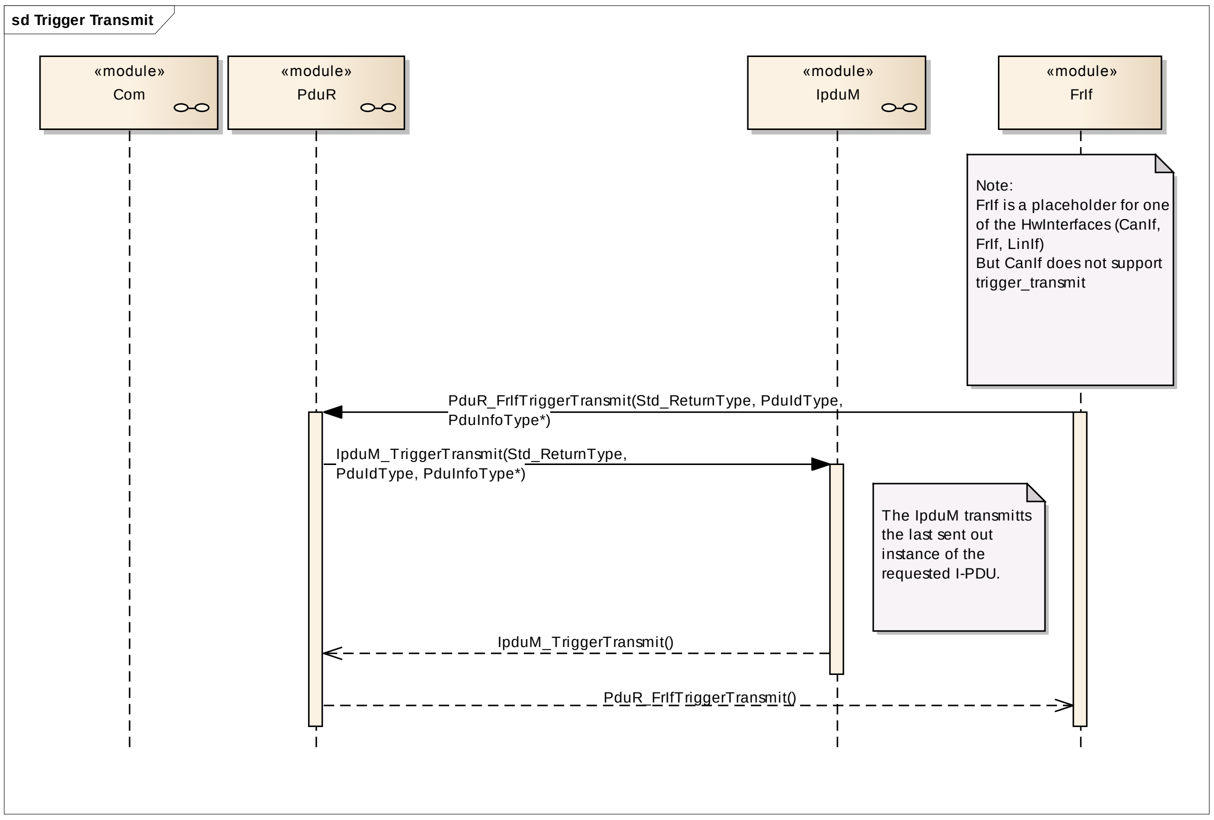
sd Trigger Transmit
«module» Com
«module» PduR
«module» IpduM
«module» FrIf
Note:
FrIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
But CanIf does not support trigger_transmit
PduR_FrIfTriggerTransmit(Std_ReturnType, PduIdType, PduInfoType*)
IpduM_TriggerTransmit(Std_ReturnType, 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, the I-PDU is allowed to be sent again.

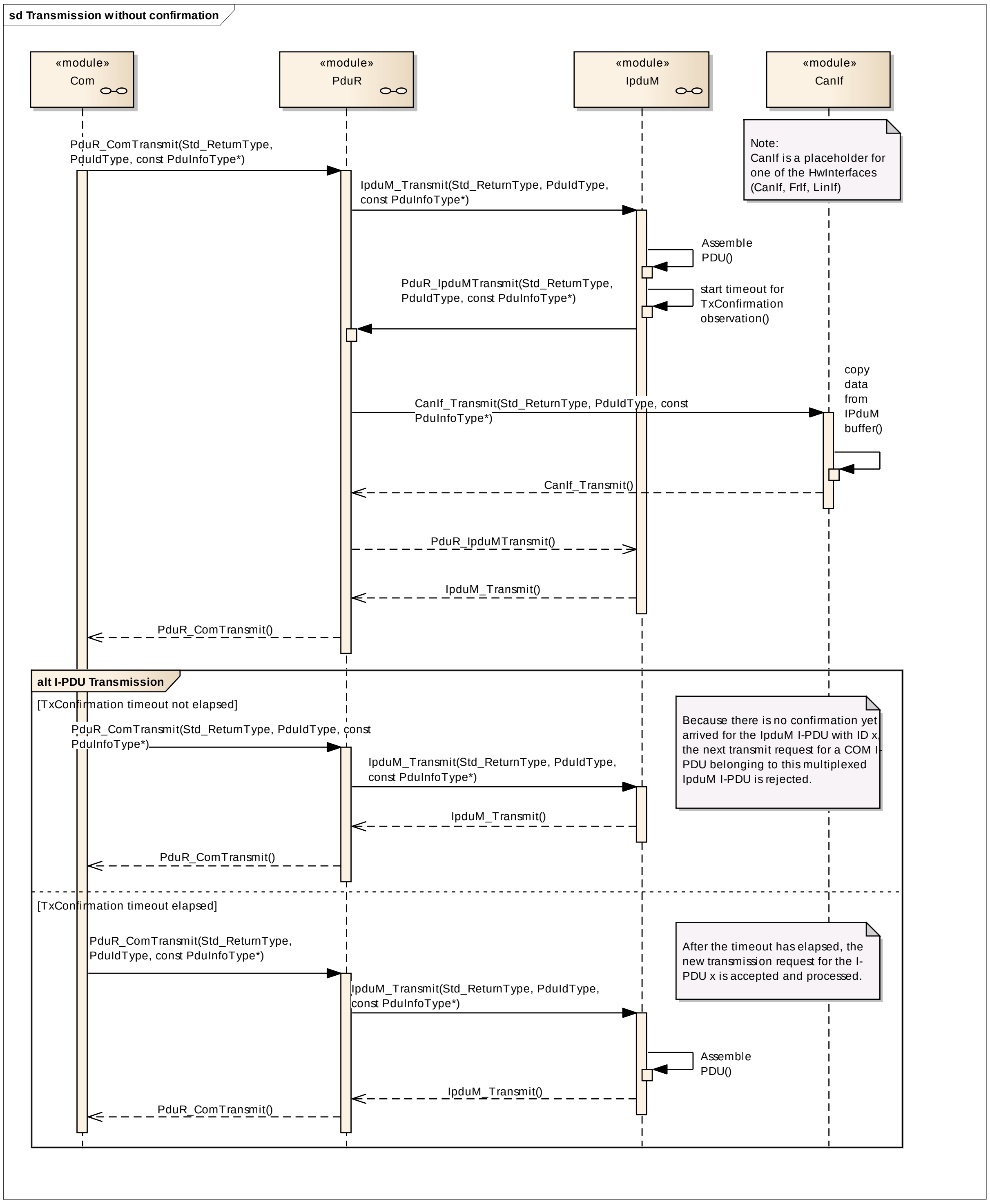
sd Transmission without confirmation
«module» Com
«module» PduR
«module» IpduM
«module» CanIf
Note: CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Assemble PDU()
start timeout for TxConfirmation observation()
PduR_IpduMTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
copy data from IPduM buffer()
CanIf_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
CanIf_Transmit()
PduR_IpduMTransmit()
IpduM_Transmit()
PduR_ComTransmit()
alt I-PDU Transmission
[TxConfirmation timeout not elapsed]
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
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*)
After the timeout has elapsed, the new transmission request for the I-PDU x is accepted and processed.
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
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. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

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

Chapter 10.3 specifies published information of the module IpduM.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [4]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration Metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:
- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

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

The IpduM module has the following three configuration variants:

- VARIANT-PRE-COMPILE
- VARIANT-LINK-TIME
- VARIANT-POST-BUILD

The VARIANT-PRE-COMPILE is designed to support the use-case where all 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

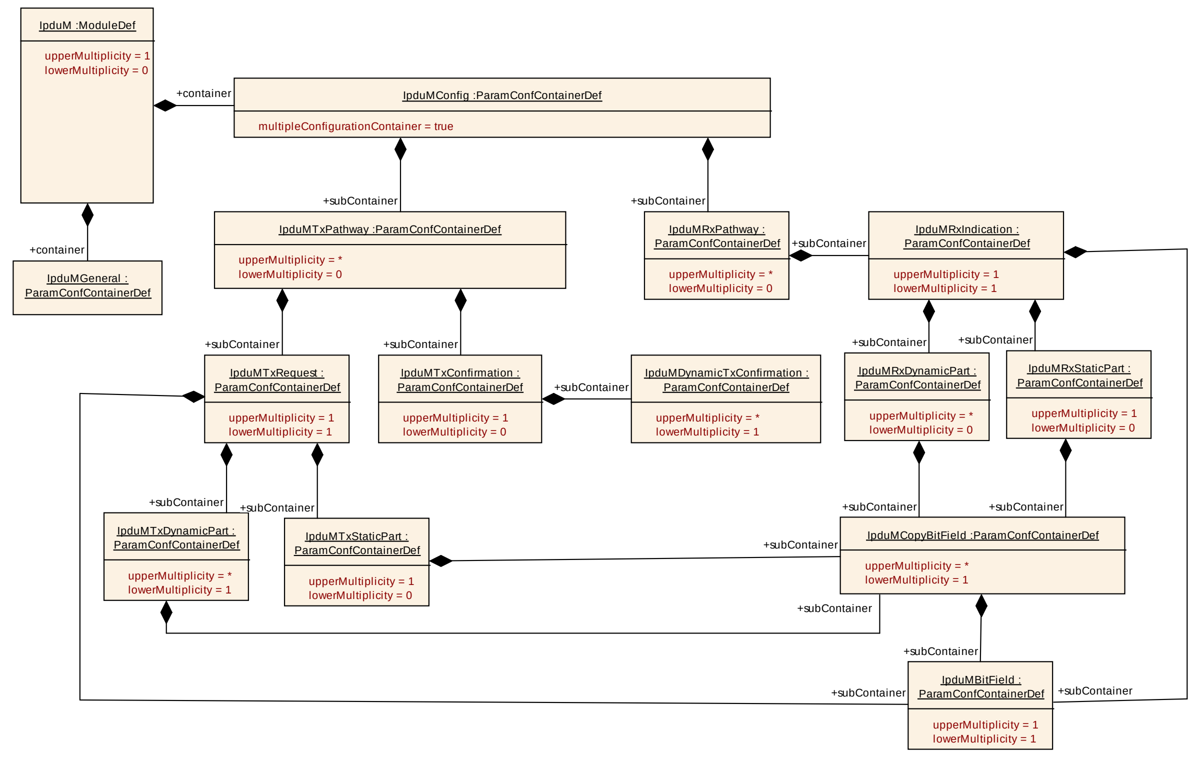

Figure 9 IpduM Configuration Overview

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.

10.2.4 IpduMGeneral

SWS Item	**IPDUM130 :**
Container Name	IpduMGeneral
Description	Contains the general configuration parameters of IpduM.
Configuration Parameters	

SWS Item	**IPDUM131 :**		
Name	IpduMConfigurationTimeBase		
Description	The period between successive ticks of AUTOSAR COM in seconds.		
Multiplicity	1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**IPDUM132 :**		
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**IPDUM133 :**
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**IPDUM134 :**		
Name	IpduMVersionInfoApi		
Description	Active/Deactivate the version information API. true: version information activated false: version information deactivated		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.5 IpduMTxPathway

SWS Item	**IPDUM070 :**
Container Name	IpduMTxPathway
Description	Contains the configuration parameters transmitted I-PDUs by the IpduM module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMTxConfirmation	0..1	configuration for a TxConfirmation
IpduMTxRequest	1	configuration for a TxRequest

10.2.6 IpduMTxRequest

SWS Item	**IPDUM052 :**
Container Name	IpduMTxRequest
Description	This is used to specify the configuration for Transmit requests. There will 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	**IPDUM162 :**
Name	IpduMByteOrder
Description	This parameter defines the ByteOrder for all IpduMSegments (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	EnumerationParamDef		
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	**IPDUM121 :**		
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	IntegerParamDef		
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	**IPDUM122 :**		
Name	IpduMInitialSelectorValue		
Description	This value is used by the initialization function to set the initial value of the selector field.		
Multiplicity	1		
Type	IntegerParamDef		
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	**IPDUM123 :**		
Name	IpduMSize		
Description	The size of the I-PDU in bytes. The maximum size is limited by the underlying communication interface. 0-8 for CAN and LIN 0-254 for FlexRay		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 254		
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	**IPDUM124 :**	
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	FloatParamDef	
Range	-INF .. INF	
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	**IPDUM125 :**		
Name	IpduMTxTriggerMode		
Description	Selects whether to send the multiplexed I-PDU immediately or at some later date.		
Multiplicity	1		
Type	EnumerationParamDef		
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	**IPDUM120 :**		
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: external		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMBitField	1	This specifies the bits that are reserved for the selector field. There can only be 1..8 bits specified.
IpduMTxDynamicPart	1..*	This (These) included container(s) must exist for each unique selector field value for this outgoing IpduM I-PDU.
IpduMTxStaticPart	0..1	This included containers configures the static part, if present.

10.2.7 IpduMTxDynamicPart

SWS Item	**IPDUM056 :**

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, the bit fields in the incoming I-PDU are packed 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.		
Configuration Parameters			

SWS Item	**IPDUM167 :**		
Name	IpduMJitUpdate		
Description	If configured to true fetch the data of this part Just-In-Time via the triggerTransmit API of the PduR.		
Multiplicity	0..1		
Type	BooleanParamDef		
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	**IPDUM127 :**		
Name	IpduMTxDynamicHandleId		
Description	This is an incoming handle id. When the handle of an incoming Tx Request matches this, the bits fields (see IpduM_CopyBitField) are copied and the IpduMTxTriggerMode is honored.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
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: External		

SWS Item	**IPDUM126 :**		
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: external		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMCopyBitField	1..*	This is a list of bit fields to copy from the incoming I-PDU to the outgoing I-PDU. This bit fields represent the subparts of the I-PDU.

10.2.8 IpduMTxStaticPart

SWS Item	**IPDUM082 :**
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, the bit fields in the incoming I-PDU are packed 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	**IPDUM167 :**		
Name	IpduMJitUpdate		
Description	If configured to true fetch the data of this part Just-In-Time via the triggerTransmit API of the PduR.		
Multiplicity	0..1		
Type	BooleanParamDef		
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	**IPDUM129 :**		
Name	IpduMTxStaticHandleId		
Description	This is an incoming handle id. When the handle of an incoming Tx Request matches this, the bits fields (see IpduMCopyBitField) are copied and the IpduMTxTriggerMode is honored.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
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: External		

SWS Item	**IPDUM128 :**		
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: external		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***

IpduMCopyBitField	1..*	Specifies the source bit fields and the destination bit position, so that the bits in the source can be copied to the bits in the destination. Within one I-PDU multiple instances of this container are used to specify the bit fields in that I-PDU. Adjacent bit fields could be merged in order to reduce the number of instances of this container.

10.2.9 IpduMTxConfirmation

SWS Item	**IPDUM050 :**
Container Name	IpduMTxConfirmation
Description	A transmit request can be confirmed by the lower layer. This container is used to generate the matching confirmations for the static and dynamic parts of a multiplexed I-PDU. When an I-PDU is transmitted by the IpduM, the selector field value in that PDU needs to be stored in the IpduM so that the confirmation for the correct dynamic part can be generated. This is state internal to the IpduM at run-time. For the purposes of this container and IpduMDynamicTxConfirmation this stored state is called Stored_Selector.
Configuration Parameters	

SWS Item	**IPDUM117 :**		
Name	IpduMStaticTxConfirmationIPduRef		
Description	This references the I-PDU to use in the TxConfirmation for the static part. This entity does not appear if there is no static part.		
Multiplicity	0..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: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMDynamicTxConfirmation	1..*	This defines the dynamic parts that also need confirmation.

10.2.10 IpduMDynamicTxConfirmation

SWS Item	**IPDUM051 :**
Container Name	IpduMDynamicTxConfirmation
Description	The dynamic part of an I-PDU can have more than one I-PDU IDs for confirmations. The correct I-PDU ID for the confirmation is found from the selector field value of a previously transmitted I-PDU. It is assumed that this selector field is stored in some internal value called Stored_Selector. When a transmit confirmation is received the Stored_Selector is used to select an instance of IpduMDynamicTxConfirmation by matching the Stored_Selector with the IpduMSelectorValue.
Configuration Parameters	

SWS Item	**IPDUM119 :**
Name	IpduMSelectorValue
Description	When the selector field of the confirmed I-PDU matches the value in here then generate a TxConfirmation for the I-PDU referenced by IpduMDynamicTxConfirmIPduRef.

Multiplicity	1		
Type	IntegerParamDef		
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	**IPDUM118 :**		
Name	IpduMDynamicTxConfirmIPduRef		
Description	This is the I-PDU ID to use in the outgoing confirmation (confirmation for the COM I-PDU) when an incoming confirmation (for an IpduM I-PDU) is received and matches the stored Stored_Selector.		
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: local		

No Included Containers

10.2.11 IpduMRxPathway

SWS Item	**IPDUM071 :**
Container Name	IpduMRxPathway
Description	Contains the configuration parameters received I-PDUs by the IpduM module.
Configuration Parameters	

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

10.2.12 IpduMRxIndication

SWS Item	**IPDUM047 :**
Container Name	IpduMRxIndication
Description	Contains the configuration for incoming RxIndication calls.
Configuration Parameters	

SWS Item	**IPDUM162 :**	
Name	IpduMByteOrder	
Description	This parameter defines the ByteOrder for all IpduMSegments (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	EnumerationParamDef	
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	**IPDUM109 :**		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
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			

SWS Item	**IPDUM108 :**		
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: external		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMBitField	1	This contains the location in the incoming I-PDU of the bit field that contains the selector field. At run-time, the selector field is used to select which dynamic part is unpacked.
IpduMRxDynamicPart	0..*	Each of these containers contains the configuration for one value of the selector field for the incoming I-PDU's dynamic part.
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 omitted.

10.2.13 IpduMRxDynamicPart

SWS Item	**IPDUM048 :**
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 of this 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.
Configuration Parameters	

SWS Item	**IPDUM113 :**		
Name	IpduMRxSelectorValue		
Description	This is the selector value that this container refers to.		
Multiplicity	1		
Type	IntegerParamDef		
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	**IPDUM112 :**		
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: external		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMCopyBitField	1..*	Contains the list of bit fields that need to be copied from the incoming I-PDU to the outgoing I-PDU.

10.2.14 IpduMRxStaticPart

SWS Item	**IPDUM049 :**
Container Name	IpduMRxStaticPart
Description	This container contains the information on how to unpack the static part of an incoming I-PDU.
Configuration Parameters	

SWS Item	**IPDUM115 :**		
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 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: external		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***

IpduMCopyBitField	1..*	Contains the list of bit fields that need to be copied from the incoming I-PDU to the outgoing I-PDU.

10.2.15 IpduMBitField

SWS Item	**IPDUM054 :**
Container Name	IpduMBitField
Description	This is used to specify a contiguous range of bits within an I-PDU. The range is inclusive.
Configuration Parameters	

SWS Item	**IPDUM110 :**		
Name	IpduMEndBit		
Description	Position of the end bit in the I-PDU.		
Multiplicity	1		
Type	IntegerParamDef		
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		

SWS Item	**IPDUM111 :**		
Name	IpduMStartBit		
Description	Position of the start bit in the I-PDU.		
Multiplicity	1		
Type	IntegerParamDef		
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 IpduMCopyBitField

SWS Item	**IPDUM053 :**
Container Name	IpduMCopyBitField
Description	Specifies the source bit fields and the destination bit position, so that the bits in the source can be copied to the bits in the destination. Within one I-PDU multiple instances of this container are used to specify the bit fields in that I-PDU. Adjacent bit fields could be merged in order to reduce the number of instances of this container.
Configuration Parameters	

SWS Item	**IPDUM114 :**	
Name	IpduMDestinationBit	
Description	Bit position in an I-PDU of the start of the destination bit field for the copy. The resulting destination field must fit inside the I-PDU.	
Multiplicity	1	
Type	IntegerParamDef	
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		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IpduMBitField	1	Source bit field.

10.2.17 IpduMConfig

SWS Item	**IPDUM059 :**
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	

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

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleID (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see 3.1 Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.