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

Document Version	1.2.0
Document Status	Final
Part of Release	2.1
Revision	20

Document Change History

Date	Version	Changed by	Change Description
07.06.2010	1.2.0	AUTOSAR Administration	• Added const to config pointer in IpduM_Init • Legal disclaimer revised
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

Disclaimer

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 Module
Dynamic part	see AUTOSAR IPDUM SRS
Static part	see AUTOSAR IPDUM SRS
Selector field	see AUTOSAR IPDUM SRS
Signal	see AUTOSAR COM SWS
Signal group	see AUTOSAR COM SWS
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 Operating System
AUTOSAR_SWS_OS,pdf

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

[7] AUTOSAR Specification of Communication
AUTOSAR_SWS_COM.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 (see AUTOSAR_COM_SWS) can not be taken into account (for more details see 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 module 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 module PDU-Router (see 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 Operating system

IPDUM079: The used OS calls are dependant on the implementation of the AUTOSAR COM module. However, the restrictions of the requirement BSW00429 in the document "General Requirements on Basic Software Modules" shall be respected.

For further information, see [AUTOSAR OS SWS].

5.2 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

5.3 File structure

5.3.1 Code file structure

IPDUM001:The code file structure shall not be defined within this specification completely. At this point, it shall be pointed out that the code-file structure shall include the following files named:

- Ipdum_Lcfg.c – for link-time configurable parameters and
- Ipdum_PBcfg.c – for post-build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.

5.3.2 Header file structure

IPDUM002:The following figure shows the include-file structure of the IPduM module.

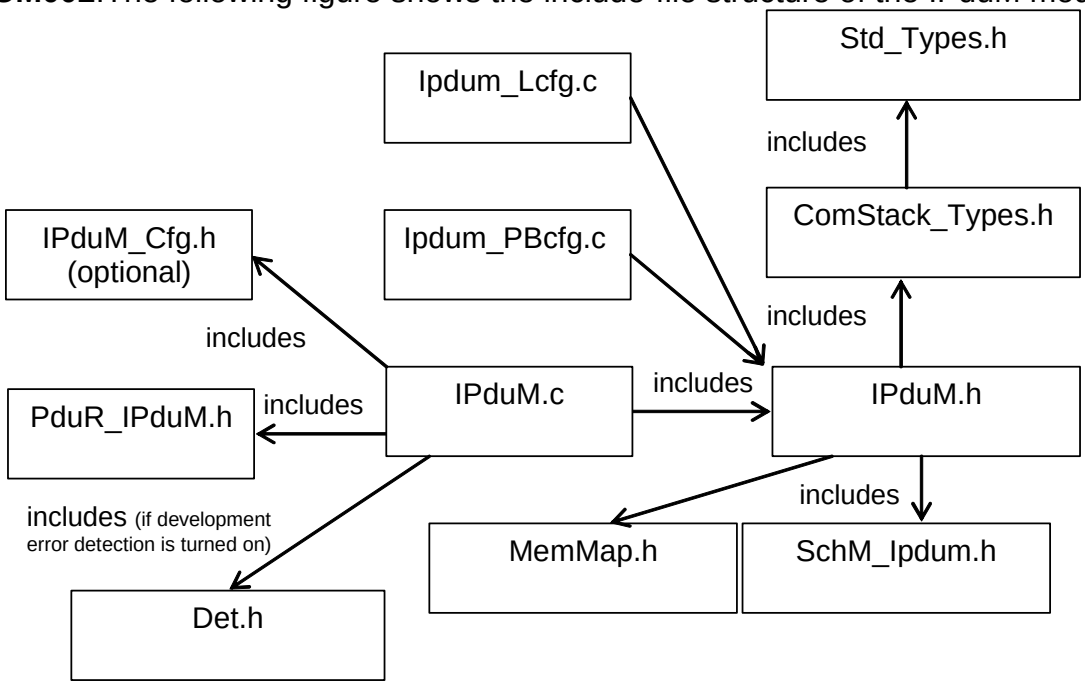


Figure 1 Header File Structure

IPDUM003:The module shall include the Dem.h file. By this inclusion, the APIs to report errors as well as the required Event ID symbols are included. This specification defines the name of the Event ID symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event ID symbols and publishes the symbols in Dem_IntErrId.h.

5.3.3 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: Direct use of compiler and platform specific keywords shall be avoided.

IPDUM075: Indicate all global data with read-only purposes by explicitly assigning the `const` keyword.

IPDUM076: It is allowed to use macros instead of functions where source code is used and runtime is critical

IPDUM077: No global data shall be defined in the header files. If global variables have to be 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, general

Requirement	*Satisfied by*
[BSW00344] Reference to link-time configuration	Chapter 10.2.2, IPDUM032
[BSW00345] Pre-compile-time configuration	Chapter 10.2.2, IPDUM059, IPDUM046, IPDUM047, IPDUM048, IPDUM049, IPDUM050, IPDUM051, IPDUM052, IPDUM053, IPDUM054, IPDUM055, 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.
[BSW101] Initialization interface	IPDUM061, IPDUM062, IPDUM032, IPDUM033, IPDUM034, IPDUM035, IPDUM036, IPDUM064, IPDUM065
[BSW003] Version identification	IPDUM037, IPDUM057, IPDUM059
[BSW004] Version check	IPDUM057, IPDUM038, IPDUM039, IPDUM059
[BSW00337] Classification of errors	IPDUM025, IPDUM026
[BSW00338] Detection and Reporting of development errors	IPDUM027, IPDUM028, IPDUM036, IPDUM059
[BSW168] Diagnostic interface	not applicable (not diagnostic interface included)
[BSW00375] Notification of wake-up reason	not applicable (this layer can not perform a wake-up)
[BSW00339] Reporting of production relevant errors and exceptions	IPDUM029, IPDUM030, IPDUM031
[BSW00369] Do not return development error codes via API	IPDUM032, IPDUM037, IPDUM040, IPDUM043, IPDUM044, IPDUM060
[BSW00336] Shutdown interface	not applicable (not needed)
[BSW00323] API parameter checking	IPDUM028
[BSW00404] Reference to post build time configuration	chapter 10.2
[BSW00405] Reference to multiple configuration sets	IPDUM032
[BSW00380] Separate C-Files for configuration parameters	IPDUM001 implementation specific
[BSW00419] Separate C-Files for pre-compile time configuration parameters	chapter 5.3 implementation specific
[BSW00381] Separate configuration header file for pre-compile time parameters	chapter 5.3 implementation specific
[BSW00412] Separate H-File for configuration parameters	chapter 5.3 implementation specific

[BSW00382] Not-used configuration elements need to be listed	not applicable (all configuration elements are used always)
[BSW00383] List dependencies of configuration files	not scope of this specification
[BSW00384] List dependencies to other modules	chapter 5
[BSW00387] Specify the configuration class of callback function	chapter 8.4
[BSW00388] Introduce containers	chapter 10.2
[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
[BSW00402] Published information	chapter 10.3
[BSW00416] Sequence of Initialization	not scope of this specification refere to Mode Management Specification.
[BSW00406] Check module initialization	IPDUM036
[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	chapter 8.5
[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 8.5
[BSW00429] Restricted BSW OS functionality access	IPDUM079

[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)
[BSW00435] and [BSW00436]	See Figure 1.
[BSW00417] Reporting of Error Events by Non-Basic Software	not applicable (this module is part of the basic software)
[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	not applicable (not scope of this specification)
[BSW162] ECU layout abstraction	not applicable (not scope of this specification)
[BSW00324] Do not use HIS I/O Library	not scope of this specification implementation specific
[BSW005] No hard coded horizontal interfaces within MCAL	not applicable (not scope of this specification)
[BSW00415] User dependent include files	figure 1
[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 8.2.1
[BSW00307] Global variables naming convention	not scope of this specification implementation specific
[BSW00310] API naming convention	chapter 8.3 and 8.4
[BSW00373]	chapter 8.5

Main processing function naming convention	
[BSW00327] Error values naming convention	IPDUM026
[BSW00335] Status values naming convention	not scope of this specification 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 8.4
[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
[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 8.2 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	not scope of this specification implementation specific
[BSW00371] Do not pass function pointers via API	chapter 8.3 and 8.4
[BSW00358] Return type of `init()` functions	chapter 8.3.1

[BSW00414] Parameter of init function	chapter 8.3.1
[BSW00376] Return type and parameters of main processing functions	chapter 8.5
[BSW00359] Return type of callback functions	chapter 8.4
[BSW00360] Parameters of callback functions	chapter 8.4
[BSW00329] Avoidance of generic interfaces	chapter 8
[BSW00330] Usage of macros / inline functions instead of functions	IPDUM076
[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	chapter 10.3
[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

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, IPDUM054
[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
[BSW02809] Initialization of multiplexed PDUs	IPDUM013, IPDUM014, IPDUM018, IPDUM069, IPDUM068, IPDUM067
[BSW02806] Semantic of the multiplexer	IPDUM010
[BSW02810] Routing of multiplexed PDUs on sender side	IPDUM063
[BSW02816] Combining of multiplexed PDUs on sender side	IPDUM015, IPDUM016, IPDUM017
[BSW02811] Triggering condition on sender side	IPDUM021, IPDUM052
[BSW02812] Routing of multiplexed PDUs on receiver side	IPDUM041, IPDUM042
[BSW02817] De-multiplexing PDUs on receiver side	IPDUM040
[BSW02813] Routing of Send Confirmations	IPDUM022, IPDUM050
[BSW02818] Confirmation replication of multiplexed PDUs	IPDUM022, IPDUM050
[BSW02814] Correct confirmation handling of multiplexed PDUs	IPDUM023, IPDUM024, IPDUM019, IPDUM020
[BSW02807] No Runtime Overhead for systems without PDU multiplexing	IPDUM012
[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 [2]).

IPDUM004:A multiplexed I-PDU shall consist 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 1.1.1).

IPDUM006:It shall be possible that the static and the dynamic part consist of more than one sub-part.

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

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

IPDUM009:The selector field of one I-PDU shall have a configurable size between one and 8 contiguous bits. (see configuration chapter 0, included container `Ipdum_BitField`)

IPDUM010:The position of the selector field within the I-PDU shall be defined by configuration (for configuration see chapter 0).

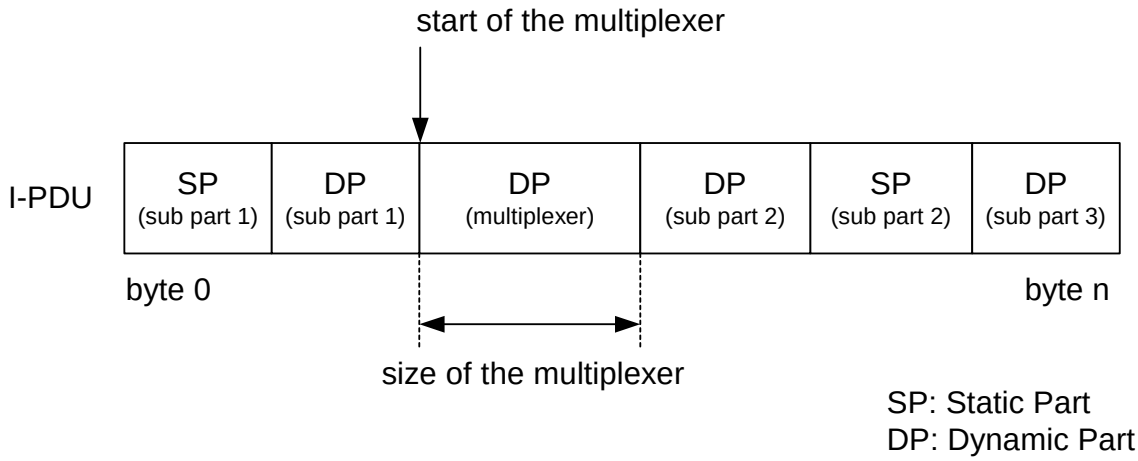


Figure 2 Possible layout of a multiplexed I-PDU

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. E.g. also if the size of a selector field with 3 bits leads to 3^2 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

IPDUM012:The Multiplexer Module IPduM is arranged next to the PDU-Router in the layered architecture of AUTOSAR [2] and
Figure 3). If no multiplexer is needed it shall be possible to build a system without this module.

IPDUM013:Each part of a multiplexed I-PDU, the static part and the different dynamic parts, shall be 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 dynamic part of one IPduM I-PDU, so the IPduM Module always combines only two I-PDUs of COM.

IPDUM014:The selector field shall not be set by IPduM module but shall be part of the COM I-PDU for the dynamic part and therefore part of the configuration of COM.

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.

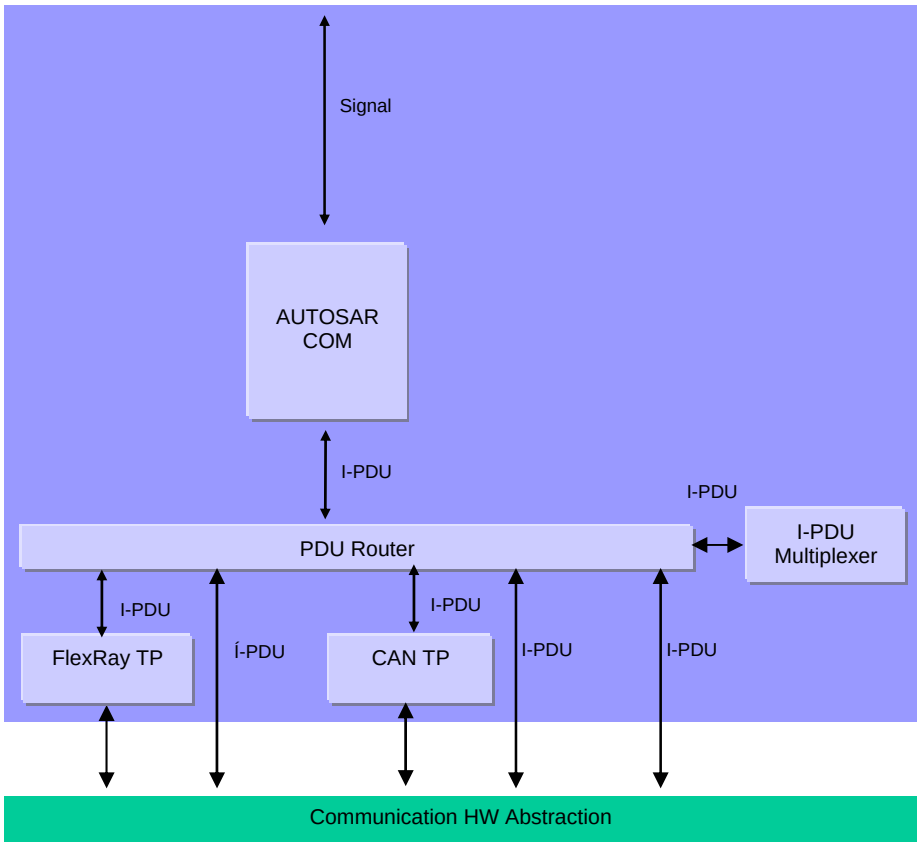


Figure 3 I-PDU Multiplexer in the Autosar Architecture

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

IPDUM061:The IPduM module shall provide an initialization function `Ipdum_Init()`, which has to be called before the first call of any other API call of the module is done (see 8.3.1).

IPDUM062:This function shall initialize all internal global variables and the buffers of the IPduM I-PDUs (for more details see chapter 8.3.1).

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 entities from the Ipdum_TxRequestConfiguration container (see chapter 0).

1) **IPDUM067**: The buffer is first filled with the pattern in `Ipdum_IpduUnusedAreasDefault`.
2) **IPDUM068**: The initial selector field `Ipdum_InitialSelectorField` 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 `Ipdum_BitField` container (see section 0) is filled with the `Ipdum_InitialSelectorField` value.

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: According to configuration of the PDU-Router module (e.g. look-up tables), the I-PDUs, which belong to multiplexed I-PDUs and represent a static or a dynamic part of a multiplexed I-PDU, shall be 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 module and have to be done by the COM layer. (For details about the timing-behavior of the new combined I-PDU see 7.4.2.)

7.4.1 Transmission request

IPDUM016: The IPduM module shall provide an `Ipdum_Transmit()` function so that the PDU-R is able to initiate the transmission of an I-PDU.

IPDUM017: When this function is called with a COM I-PDU, the function shall assemble the related IPduM I-PDU (using the related static and dynamic part) and transmit it according to the trigger conditions (see chapter 7.4.2).

IPDUM018: An initial value for the outgoing I-PDU shall be defined 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 (see chapter 7.3 and chapter 0).

IPDUM019: A dedicated timeout shall be configured for each IPduM I-PDU within the IPduM module. 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 this timeout is not elapsed and no transmission confirmation for an IPduM I-PDU is received a new transmission request from the upper layer with an COM I-PDUs that belongs to the same IPduM I-PDUs is not allowed and returned 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 [AUTOSAR_COM_SWS]).

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: It shall be configurable whether the IPduM module sends 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 [AUTOSAR_COM_SWS])

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 a I-PDU is only sent out because of a TriggerTransmit of an lower layer.

IPDUM081: With the API Ipdum_TriggerTransmit it shall be possible for lower layers to trigger a send out of an I-PDU,

7.4.3 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 module receives a TxConfirmation for a specific IPduM I-PDU, it shall translate this confirmation into two confirmations for the COM I-PDUs, which were contained in the last sent out multiplexed IPduM I-PDU.

IPDUM023: If the TxConfirmation is not received within the configured timeout (see chapter 7.4.1 and for configuration see chapter 0) the IPduM shall allow new transmission requests for this specific I-PDU after timeout is elapsed.

IPDUM024: Unexpected TxConfirmations shall be silently discarded. (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.

Note:
Depending on the configuration (see 0) there are zero, one or two confirmations given to COM for one send request.

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.

For requirements description see chapter 8.3.3.

7.6 Error classification

IPDUM025: Values for production code Event IDs are assigned externally by the configuration of the DEM. They are published in the file Dem_IntErrId.h and included via Dem.h.

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`
Transmit request is not accepted	Production	`E_NOT_OK`	`assigned externally`

7.7 Error detection

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

IPDUM028: If the `IPDUM_DEV_ERROR_DETECT` 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.

7.8 Error notification

IPDUM030: Detected development errors shall be reported to the error hook of the Development Error Tracer (DET) if the pre-processor switch `IPDUM_DEV_ERROR_DETECT` is set (see chapter 10).

IPDUM031: Production errors shall be reported to Diagnostic Event Manager.

8 API specification

8.1 Imported types

8.1.1 Standard types

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

- Std_Types.h
- ComStack_Types.h

The following types are used:

- Std_ReturnType
- Std_VersionInfoType
- PduIdType
- PduInfoType

8.2 Type definitions

8.2.1 Ipdum_ConfigType

Type:	Structure
Range:	The content of the initialization data structure is 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 (out):	none
Return value:	none
Description:	**IPDUM033**: Ipdum_Init() shall initialize all module related global variables. **IPDUM034**: Ipdum_Init() shall initialize all I-PDUs with the default values. **IPDUM035**:

	The default value of the selector field shall be initialized by the Ipdum_Init() function with a configurable value. (see configuration chapter) **IPDUM064**: The states of the timeout monitors shall be initialized. **IPDUM065**: The state of the TxConfirmation IDs shall be initialized. **IPDUM036**: If the config parameter does not correspond to a valid configuration then the development error `IPDUM_E_PARAM` is generated. The behavior of the IPduM is unspecified until a correct call to Ipdum_Init() is made.
Caveats:	None
Configuration:	see chapter 10.2

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 (out):	`versioninfo` Pointer to where to store the version information of this module.
Return value:	`None`
Description:	**IPDUM038**: This service returns the version information of this module. The version information includes: - Module ID - Vendor ID - Vendor specific version numbers (BSW00407). **IPDUM039**: This function shall be pre compile time configurable `On/Off` by the configuration parameter: `IPDUM_VERSION_INFO_API` Note: If source code for caller and called of this function is available, this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	None
Configuration:	see 0

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:	Re-entrant for different I-PDUs. Non re-entrant for the same I-PDU.	
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 (out):	`None`	
Return value:	`Std_ReturnType` `E_OK`	Transmit request is accepted
	`E_NOT_OK`	Transmit request is not accepted
Description:	This function is called by the PDU-Router to request a transmission. For a detailed description read chapter 7.4.1	
Caveats:	None	
Configuration:	see chapter 10.2	

8.4 Call-back notifications

8.4.1 Ipdum_RxIndication

IPDUM040:

Service name:	Ipdum_RxIndication	

Syntax:	`void Ipdum_RxIndication(`
	`  PduIdType  PdumRxPduId,`
	`  const uint8 *SduPtr`
	`)`	

Service ID [hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non-reentrant	
Parameters (in):	`PdumRxPduId`	ID of I-PDU that has been received.
	`SduPtr`	Pointer to underlying layer's L-SDU (buffer of received payload). This pointer could point to a CAN, LIN for FlexRay SDU.
Parameters (out):	`None`	
Return value:	`None`	
Description:	This function 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 this function 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 this function 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).
Caveats:	This function might be called in interrupt context (e.g. from receive interrupt)
Configuration:	see chapter 10.2

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
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 (out):	none
Return value:	none
Description:	This function is called by the lower layer after the I-PDU has been transmitted on the network. This function 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 which has to route them to COM.
Caveats:	This function might be called in interrupt context (e.g. from transmit interrupt)
Configuration:	see chapter 10.2

8.4.3 Ipdum_TriggerTransmit

IPDUM060:

Service name:	Ipdum_TriggerTransmit

Syntax:	`void Ipdum_TriggerTransmit (`
	`  PduIdType  PdumTxPduId,`
	`  uint8  *SduPtr`
	`)`

Service ID [hex]:	0x05
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	`PdumTxPduId` ID of IPduM I-PDU that is requested to be transmitted by IPduM.
Parameters (out):	`SduPtr` Pointer to transmit buffer of I-PDU.
Return value:	None
Description:	This function is called by the lower layer when an IPduM I-PDU shall be transmitted. Within this function, the IPduM module shall copy the contents of its I-PDU transmit buffer to the I-PDU buffer given by SduPtr.

	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).
Caveats:	This function might be called in interrupt context.
Configuration:	see chapter 10.2

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.

Service name:	Ipdum_MainFunction
Service ID [hex]:	0x10
Description:	This function shall perform the processing of the IpduM activities that are not directly initiated by the calls from PDU-R. At least the TxConfirmation time observation has to be done within this service
Timing:	fixed cyclic with pre condition
Pre condition:	IpduM module is waiting for a TxConfirmation
Configuration:	**IPDUM072**: Cycle time is dependent on the `IPDUM_TX_CONFIRMATION_TIMEOUT.` Its value shall be the smallest configured `IPDUM_TX_CONFIRMATION_TIMEOUT.`

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.

API function	*Module*	*Description*
`PduR_IpdumTxConfirmation`	PDU-R	Confirm the transmission of an I-PDU.
`PduR_IpdumRxIndication`	PDU-R	Indicate the reception of an I-PDU.
`PduR_IpdumTransmit`	PDU-R	Request the transmission of I-PDU.

8.6.2 Optional Interfaces

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

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
`Det_ReportError`	Det	Development error notification	IPDUM_DEV_ERROR_DETECT

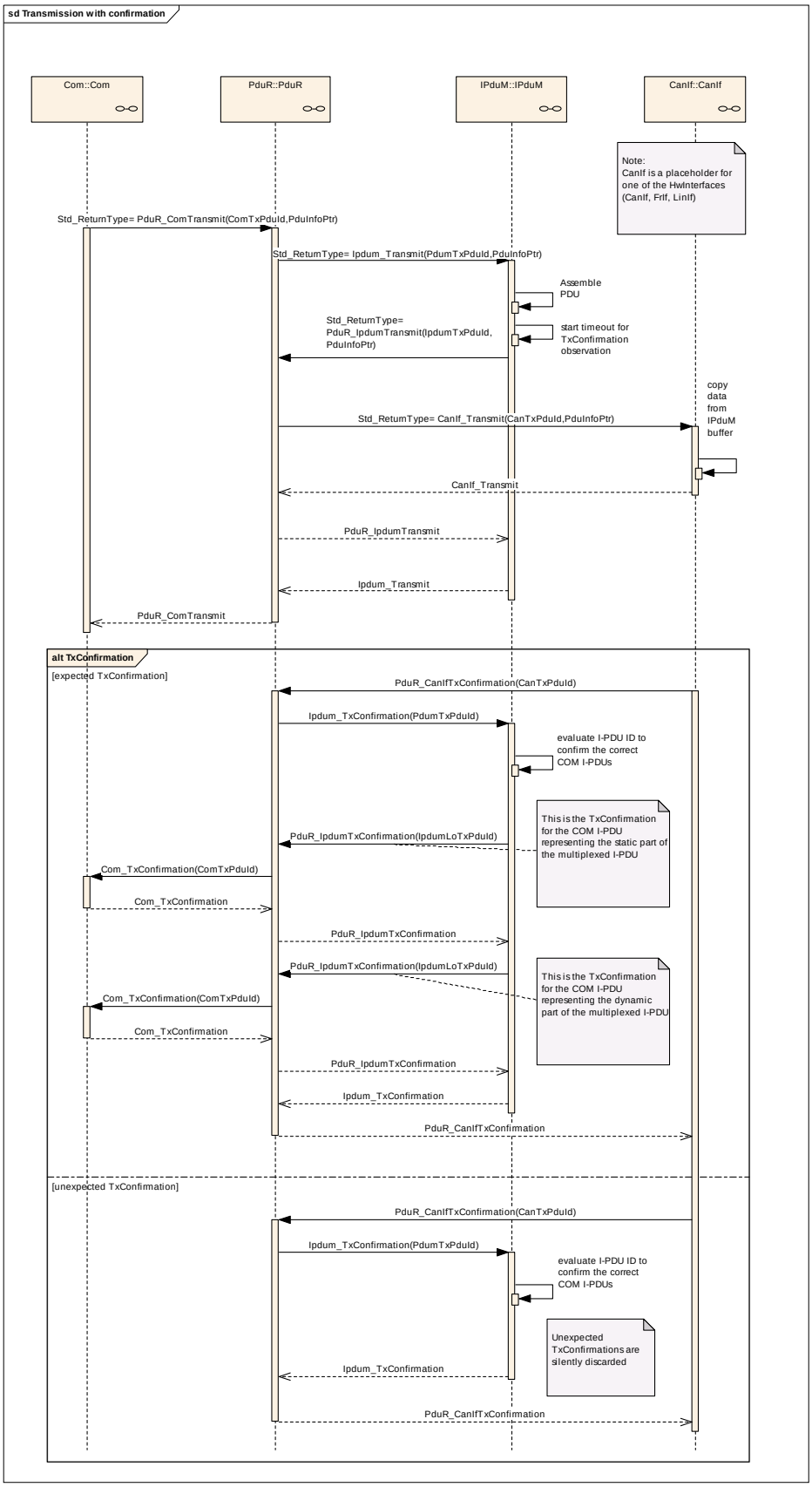
8.6.3 Configurable interfaces

Not applicable

9 Sequence diagrams

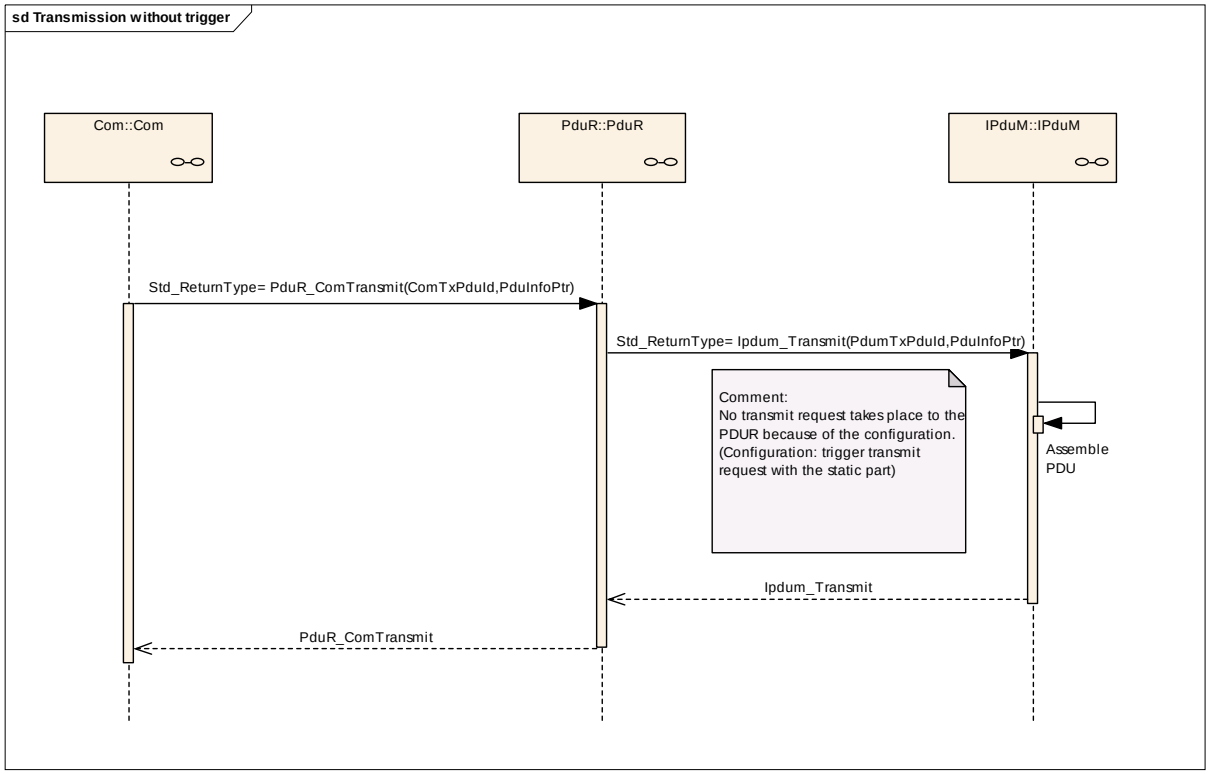
9.1 Transmission of a multiplexed I-PDU and Transmit confirmation

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
Com::Com
PduR::PduR
IPduM::IPduM
CanIf::CanIf
Note:
CanIf is a placeholder for one of the HwInterfaces (CanIf, FrIf, LinIf)
Std_ReturnType= PduR_ComTransmit(ComTxPduId,PduInfoPtr)
Std_ReturnType= Ipdum_Transmit(PdumTxPduId,PduInfoPtr)
Assemble PDU
Std_ReturnType= PduR_IpdumTransmit(IpdumTxPduId, PduInfoPtr)
start timeout for TxConfirmation observation
copy data from IPduM buffer
Std_ReturnType= CanIf_Transmit(CanTxPduId,PduInfoPtr)
CanIf_Transmit
PduR_IpdumTransmit
Ipdum_Transmit
PduR_ComTransmit
alt TxConfirmation
[expected TxConfirmation]
PduR_CanIfTxConfirmation(CanTxPduId)
Ipdum_TxConfirmation(PdumTxPduId)
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(IpdumLoTxPduId)
Com_TxConfirmation(ComTxPduId)
Com_TxConfirmation
PduR_IpdumTxConfirmation
PduR_IpdumTxConfirmation(IpdumLoTxPduId)
This is the TxConfirmation for the COM I-PDU representing the dynamic part of the multiplexed I-PDU
Com_TxConfirmation(ComTxPduId)
Com_TxConfirmation
PduR_IpdumTxConfirmation
Ipdum_TxConfirmation
PduR_CanIfTxConfirmation
[unexpected TxConfirmation]
PduR_CanIfTxConfirmation(CanTxPduId)
Ipdum_TxConfirmation(PdumTxPduId)
evaluate I-PDU ID to confirm the correct COM I-PDUs
Unexpected TxConfirmations are silently discarded
Ipdum_TxConfirmation
PduR_CanIfTxConfirmation

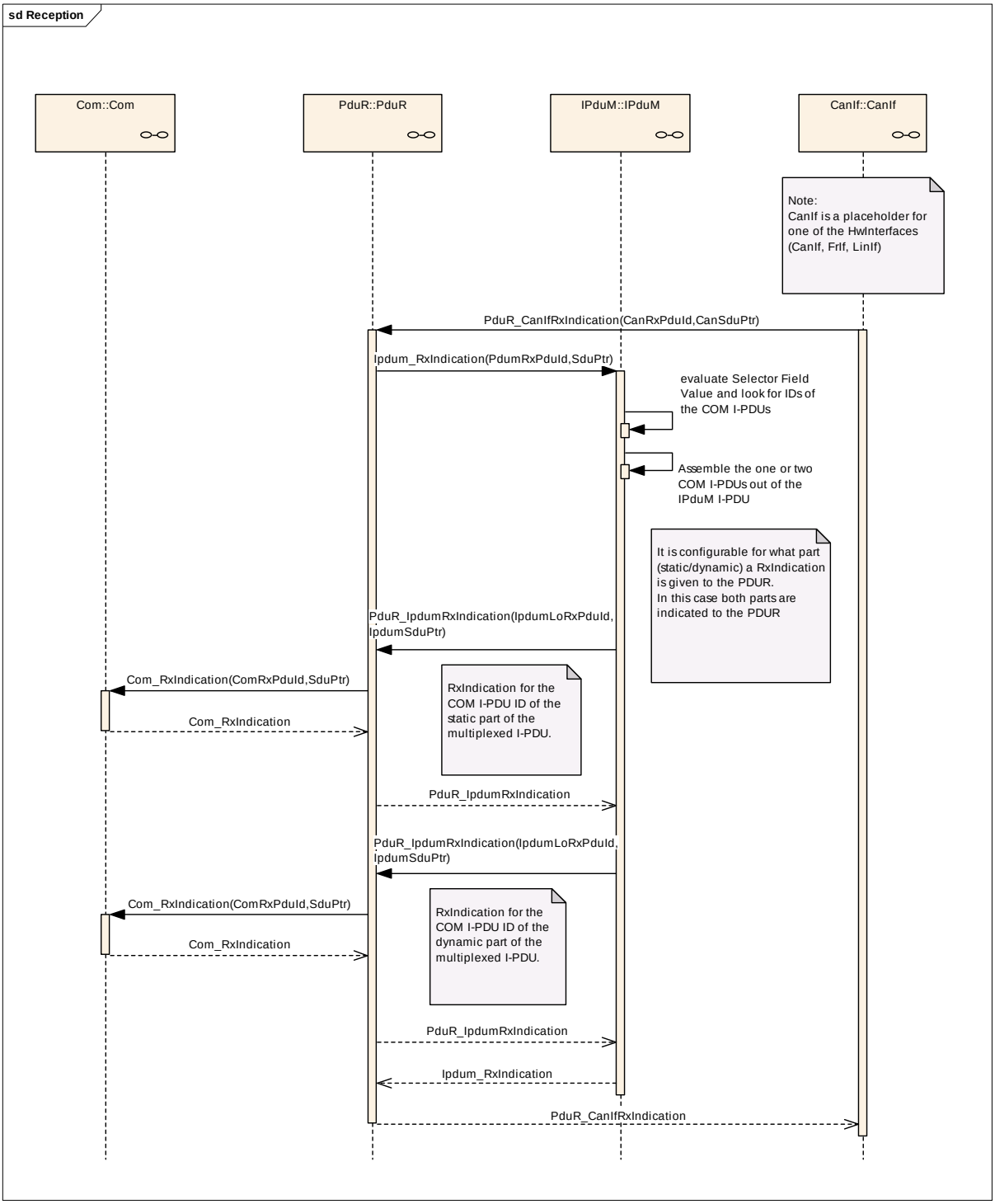
9.2 Transmission of a multiplexed I-PDU without Trigger

The following sequence chart shows a transmit request initiated by the COM layer. Because of the configuration of the IPduM module no transmit request for the IPduM I-PDU takes place (for configuration see IPDUM052).



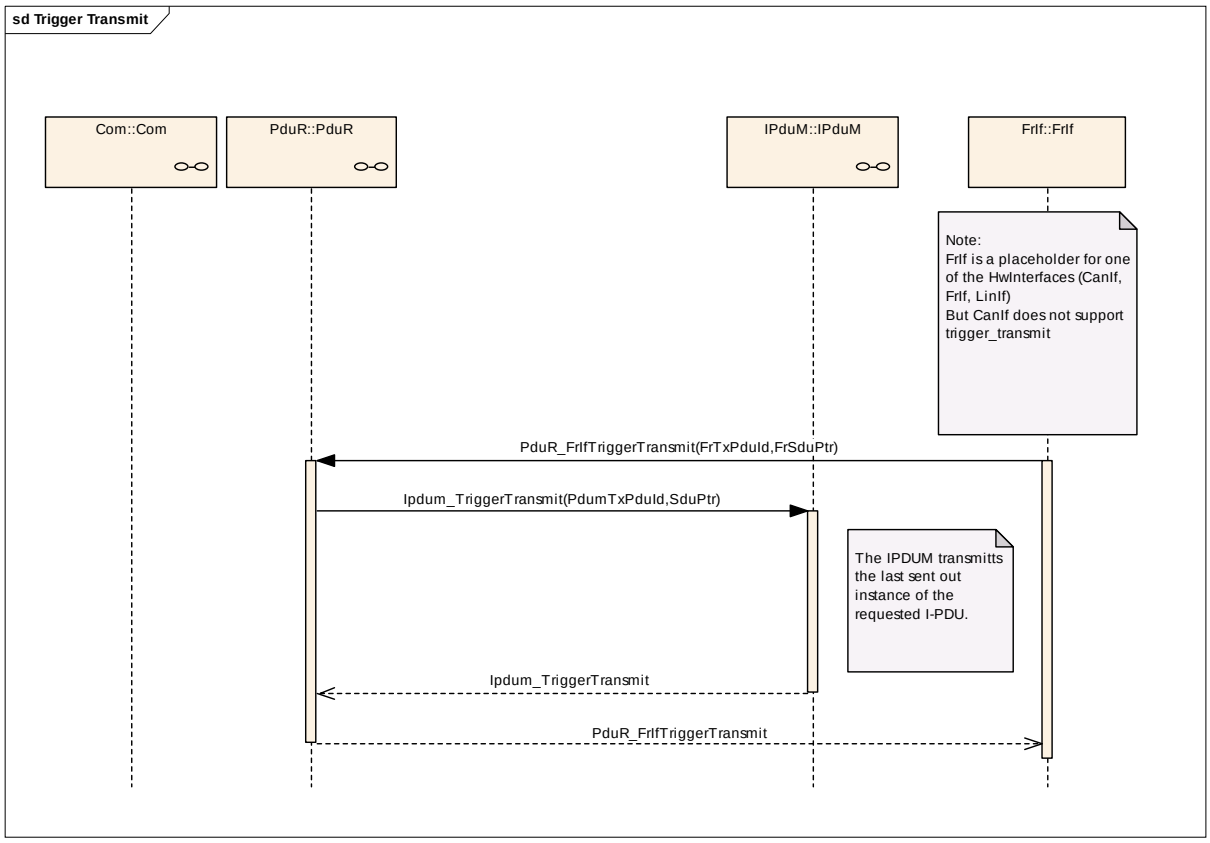
9.3 Reception of the multiplexed I-PDU

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 a RxIndication to the PDU-R module (for configuration see 0).



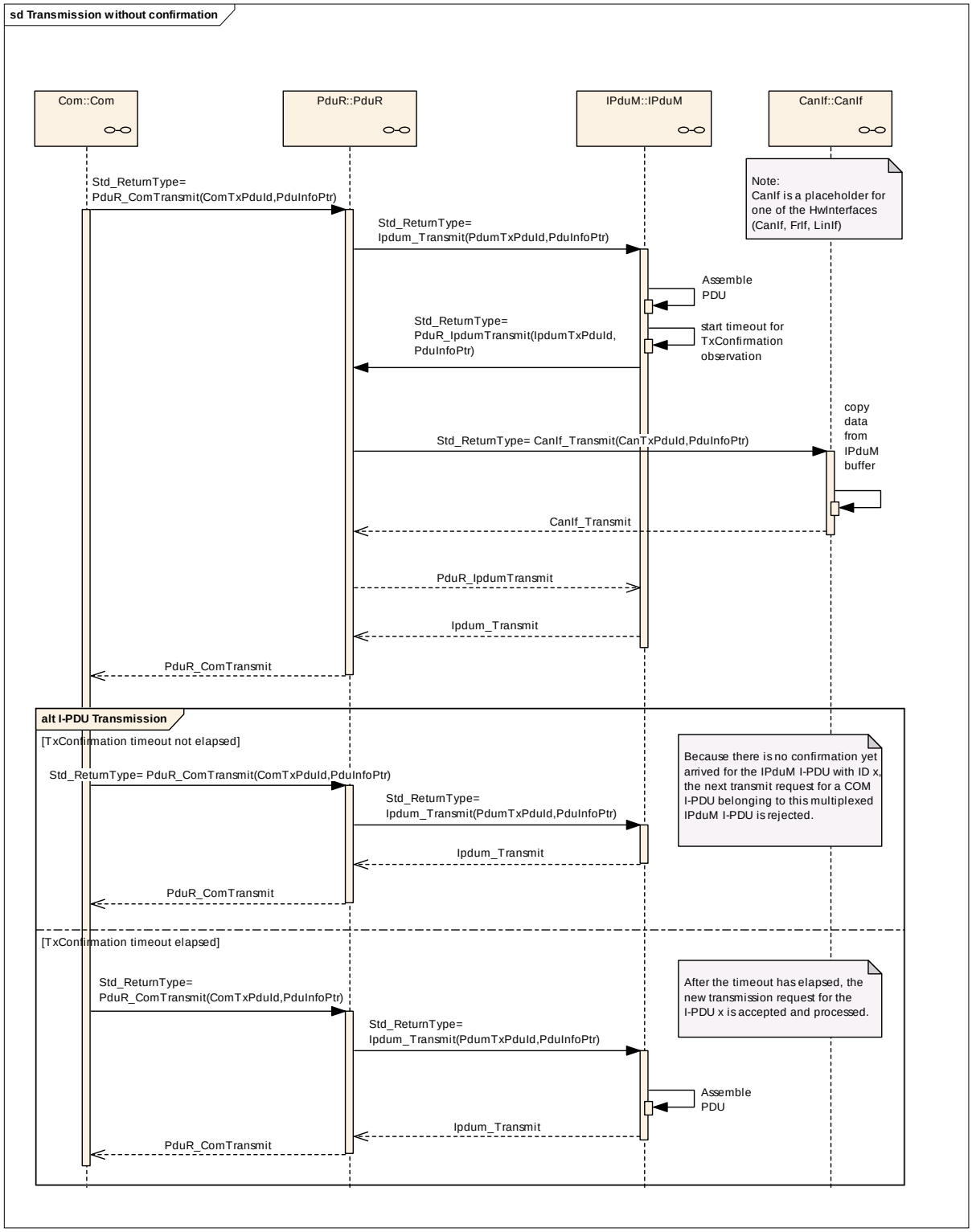
9.4 Trigger Transmit

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



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 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

Thus describe the possible configuration variants of this module. Each Variant must have a unique name which could be referenced to in later chapters. The maximum number of allowed variants is 3.

10.1.3 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.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

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

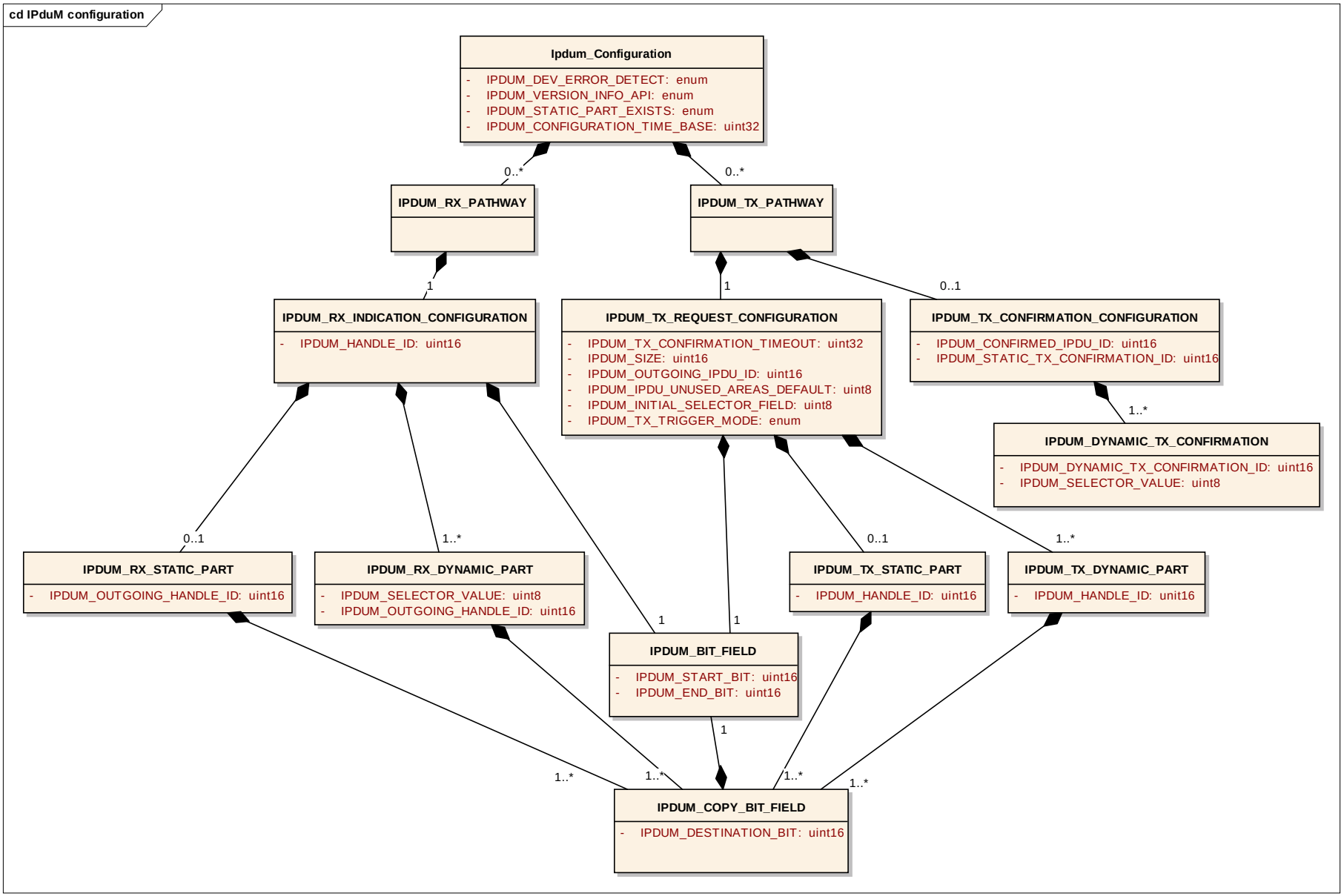
There are two variants called: Fixed and Reconfigurable.

The Fixed variant 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 Reconfigurable variant 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

Hint:
Please pay attention:
The writing of the names of the parameters in the following picture does not correspond with the following description. All names shall be in capital letters with underscores between the words, as done in the chapters starting at 10.2.3.


cd IPduM configuration
Ipdum_Configuration
- IPDUM_DEV_ERROR_DETECT: enum
- IPDUM_VERSION_INFO_API: enum
- IPDUM_STATIC_PART_EXISTS: enum
- IPDUM_CONFIGURATION_TIME_BASE: uint32
IPDUM_RX_PATHWAY
IPDUM_TX_PATHWAY
IPDUM_RX_INDICATION_CONFIGURATION
- IPDUM_HANDLE_ID: uint16
IPDUM_TX_REQUEST_CONFIGURATION
- IPDUM_TX_CONFIRMATION_TIMEOUT: uint32
- IPDUM_SIZE: uint16
- IPDUM_OUTGOING_IPDU_ID: uint16
- IPDUM_IPDU_UNUSED_AREAS_DEFAULT: uint8
- IPDUM_INITIAL_SELECTOR_FIELD: uint8
- IPDUM_TX_TRIGGER_MODE: enum
IPDUM_TX_CONFIRMATION_CONFIGURATION
- IPDUM_CONFIRMED_IPDU_ID: uint16
- IPDUM_STATIC_TX_CONFIRMATION_ID: uint16
IPDUM_DYNAMIC_TX_CONFIRMATION
- IPDUM_DYNAMIC_TX_CONFIRMATION_ID: uint16
- IPDUM_SELECTOR_VALUE: uint8
IPDUM_RX_STATIC_PART
- IPDUM_OUTGOING_HANDLE_ID: uint16
IPDUM_RX_DYNAMIC_PART
- IPDUM_SELECTOR_VALUE: uint8
- IPDUM_OUTGOING_HANDLE_ID: uint16
IPDUM_TX_STATIC_PART
- IPDUM_HANDLE_ID: uint16
IPDUM_TX_DYNAMIC_PART
- IPDUM_HANDLE_ID: unit16
IPDUM_BIT_FIELD
- IPDUM_START_BIT: uint16
- IPDUM_END_BIT: uint16
IPDUM_COPY_BIT_FIELD
- IPDUM_DESTINATION_BIT: uint16
0..*
0..*
1
1
0..1
1..*
0..1
1..*
0..1
1..*
1
1
1
1..*
1..*
1..*
1..*

10.2.3 IPDUM_CONFIGURATION

SWS Item	**IPDUM059**:
Container Name	IPDUM_CONFIGURATION
Description	This container contains the general configuration parameters of IPduM
Configuration Parameters	

Name	IPDUM_DEV_ERROR_DETECT		
Description	Active/Deactivate the detection of development errors, for production code this parameter has to be OFF		
Type	Enumeration		
Unit	--		
Range	ON OFF	error detection activated error detection deactivated	
Configuration Class	***Pre-compile***	x	Valid variants: Fixed, Reconfigurable.
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	IPDUM_VERSION_INFO_API		
Description	Active/Deactivate the version information API (see 8.3.2)		
Type	Enumeration		
Unit	--		
Range	ON OFF	version information activated version information deactivated	
Configuration Class	***Pre-compile***	x	Valid variants: Fixed, Reconfigurable.
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	IPDUM_STATIC_PART_EXISTS		
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 “NO” then it will not be possible to add static parts after compilation.		
Type	Enumeration		
Unit	--		
Range	YES NO	A static part may exist. A static part will never exist.	
Configuration Class	***Pre-compile***	x	Valid variants: Fixed, Reconfigurable.
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Name	IPDUM_CONFIGURATION_TIME_BASE	
Description	The period between successive “ticks” of AUTOSAR COM in seconds.	
Type	Real	
Unit	--	
Range	--	--

Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	--	--
Scope	local		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IPDUM_TX_PATHWAY	0..n	includes information about sent I-PDUs
IPDUM_RX_PATHWAY	0..n	includes information about received I-PDUs

10.2.4 IPDUM_TX_PATHWAY

SWS Item	**IPDUM070**:
Container Name	IPDUM_TX_PATHWAY
Description	This container contains the configuration parameters transmitted I-PDUs by the IPduM module.
Configuration Parameters	

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

10.2.5 IPDUM_RX_PATHWAY

SWS Item	**IPDUM071**:
Container Name	IPDUM_RX_PATHWAY
Description	This container contains the configuration parameters received I-PDUs by the IPduM module.
Configuration Parameters	

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

10.2.6 IPDUM_RX_INDICATION_CONFIGURATION

SWS Item	**IPDUM047**:
Container Name	IPDUM_RX_INDICATION_CONFIGURATION
Description	This container contains the configuration for incoming RxIndication calls.
Configuration Parameters	

Name	IPDUM_HANDLE_ID		
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.		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	external		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IPDUM_BIT_FIELD	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.
IPDUM_RX_STATIC_PART	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.
IPDUM_RX_DYNAMIC_PART	1..n	Each of these containers contains the configuration for one value of the selector field for the incoming I-PDU's dynamic part.

10.2.7 IPDUM_RX_DYNAMIC_PART

SWS Item	**IPDUM048**:
Container Name	IPDUM_RX_DYNAMIC_PART
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 IPDUM_SELECTOR_VALUE the I-PDU is unpacked according to the values in the IPDUM_COPY_BITFIELD and then the new I-PDU constructed and sent out with the I-PDU ID in IPDUM_OUTGOING_HANDLE_ID
Configuration Parameters	

Name	IPDUM_SELECTOR_VALUE		
Description	This is the selector value that this container refers to.		
Type	uint8		
Unit	byte		
Range	0..0xff		
Configuration Class	***Pre-compile***	--	--
	Link time	X	Valid variants: Fixed.
	Post Build	X	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_OUTGOING_HANDLE_ID		
Description	When the new I-PDU is sent out it is sent with this I-PDU ID.		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	external		
Dependency	--		

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

10.2.8 IPDUM_RX_STATIC_PART

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

Name	IPDUM_OUTGOING_HANDLE_ID		
Description	When the new I-PDU is sent out it is sent with this I-PDU ID.		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	X	Valid variants: Fixed.
	Post Build	X	Valid variants: Reconfigurable.
Scope	external		
Dependency	--		

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

10.2.9 IPDUM_TX_CONFIRMATION_CONFIGURATION

SWS Item	**IPDUM050**:
Container Name	IPDUM_TX_CONFIRMATION_CONFIGURATION
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 IPDUM_DYNAMIC_CONFIRMATION this stored state is called Stored_Selector
Configuration Parameters	

Name	IPDUM_CONFIRMED_IPDU_ID		
Description	This is the I-PDU ID of the previously transmitted I-PDU that is to be confirmed. Therefore, the value of this entity must match one of the set of IPDUM_OUTGOING_IPDU_ID in IPDUM_TX_REQUEST_CONFIGURATION. This is the ID of the IPduM I-PDU.		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	Local		
Dependency	--		

Name	IPDUM_STATIC_TX_CONFIRMATION_ID		
Description	This is the I-PDU ID to use in the TxConfirmation for the static part. It is a COM I-PDU ID.		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	This entity does not appear if there is no static part.		

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

10.2.10 IPDUM_DYNAMIC_TX_CONFIRMATION

SWS Item	**IPDUM051**:
Container Name	IPDUM_DYNAMIC_TX_CONFIRMATION
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 IPDUM_DYNAMIC_CONFIRMATION by matching the Stored_Selector with the IPDUM_SELECTOR_VALUE
Configuration Parameters	

Name	IPDUM_SELECTOR_VALUE		
Description	When the selector field of the confirmed I-PDU matches the value in here then generate a TxConfirmation with the I-PDU ID indicated using the IPDUM_CONFIRMATION_ID value.		
Type	uint8		
Unit	byte		
Range	0..0xff		
Configuration Class	***Pre-compile***	--	--
	Link time	X	Valid variants: Fixed.
	Post Build	X	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_DYNAMIC_TX_CONFIRMATION_ID		
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."		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	X	Valid variants: Fixed.
	Post Build	X	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

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

10.2.11 IPDUM_TX_REQUEST_CONFIGURATION

SWS Item	**IPDUM052**:
Container Name	IPDUM_TX_REQUEST_CONFIGURATION
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	

Name	IPDUM_TX_CONFIRMATION_TIMEOUT		
Description	This timeout 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.		
Type	float		
Unit	seconds		
Range	Target dependent		
Configuration Class	***Pre-compile***	--	
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_SIZE		
Description	Size of the outgoing I-PDU in bits. There is no use case for this being larger than 64 bits. Therefore the maximum size is 64 bits.		
Type	uint16		
Unit	bits		
Range	1..64		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_OUTGOING_IPDU_ID		
Description	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.		
Type	uint16		
Unit	--		
Range	0..n		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	external		
Dependency	--		

Name	IPDUM_IPDU_UNUSED_AREAS_DEFAULT		
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.		
Type	uint8		
Unit	byte		
Range	0..0xff		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_INITIAL_SELECTOR_VALUE		
Description	This value is used by the initialization function to set the initial value of the selector field.		
Type	Uint8		
Unit	byte		
Range	0..0xff		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_TX_TRIGGER_MODE		
Description	Selects whether to send the multiplexed I-PDU immediately or at some later date.		
Type	Enum		
Unit	--		
Range	IPDUM_NONE	Only the buffer in the IPduM are written but not send is triggered, used for IPduM I-PDUs which are requested by TriggerTransmit.	
	IPDUM_STATIC_PART_TRIGGER	Writing the I-PDU representing the static part does trigger a sending of the I-PDU	
	IPDUM_DYNAMIC_PART_TRIGGER	Writing the I-PDU representing the dynamic part does trigger a sending of the I-PDU	
	IPDUM_STATIC_OR_DYNAMIC_PART_TRIGGER	Writing the I-PDU representing the static or the dynamic part does trigger a sending of the I-PDU	
Configuration Class	***Pre-compile***	--	--
	Link time	X	Valid variants: Fixed.
	Post Build	X	Valid variants: Reconfigurable.
Scope	Local		
Dependency	--		

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

10.2.12 IPDUM_TX_DYNAMIC_PART

SWS Item	**IPDUM056**:
Container Name	IPDUM_TX_DYNAMIC_PART
Description	Configuration parameters for an instance of a Tx_Request call into the IPduM. When a Tx Request with the IPDUM_HANDLE_ID 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 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	

Name	IPDUM_HANDLE_ID		
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 `Ipdum_TxTriggerMode` is honored.		
Type	Uint16		
Unit	--		
Range	`1..n`		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	External		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
`IPDUM_COPY_BIT_FIELD`	`1..n`	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. See chapter 7.1.

10.2.13 IPDUM_TX_STATIC_PART

SWS Item	**IPDUM082**:
Container Name	IPDUM_TX_STATIC_PART
Description	Configuration parameters for an instance of a Tx_Request call into the IPduM. When a Tx Request with the IPDUM_HANDLE_ID 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	

Name	IPDUM_HANDLE_ID		
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 `Ipdum_TxTriggerMode` is honored.		
Type	Uint16		
Unit	--		
Range	`1..n`		
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	External		

Dependency	--

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IPDUM_COPY_BIT_FIELD	1..n	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. See chapter 7.1.

10.2.14 IPDUM_COPY_BIT_FIELD

SWS Item	**IPDUM053**:
Container Name	IPDUM_COPY_BIT_FIELD
Description	Two bit fields are specified, a source and a destination, 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	

Name	IPDUM_DESTINATION_BIT		
Description	Bit position in an I-PDU of the start of the bit field that is the destination for the copy.		
Type	Uint8		
Unit	bits		
Range	0..63	Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit	
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

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

10.2.15 IPDUM_BIT_FIELD

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

Name	IPDUM_START_BIT
Description	Bit position in an I-PDU of the start of the bit field
Type	uint8
Unit	bits

Range	0..63	Value must fit inside the I-PDU. Value must be the same as or lower than Ipdum_EndBit	
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

Name	IPDUM_END_BIT		
Description	Bit position in an I-PDU of the end of the bit field		
Type	uint8		
Unit	bits		
Range	0..63	Value must fit inside the I-PDU. Value must be the same as or higher than Ipdum_StartBit	
Configuration Class	***Pre-compile***	--	--
	Link time	x	Valid variants: Fixed.
	Post Build	x	Valid variants: Reconfigurable.
Scope	local		
Dependency	--		

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

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.

SWS Item	**IPDUM057**:	
Information elements		
Information element name	***Type / Range***	***Information element description***
IPDUM_VENDOR_ID	#define/ uint16	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
IPDUM_MODULE_ID	#define/ uint8	Module ID of this module from Module List
IPDUM_AR_MAJOR_VERSION	#define/ uint8	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
IPDUM_AR_MINOR_VERSION	#define/ uint8	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
IPDUM_AR_PATCH_VERSION	#define/ uint8	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
IPDUM_SW_MAJOR_VERSION	#define/ uint8	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
IPDUM_SW_MINOR_VERSION	#define/ uint8	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
IPDUM_SW_PATCH_VERSION	#define/ uint8	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.