| Document Title | Specification of I-PDU Multi-plexer |
|---|---|
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 182 |
| **Document Classification** | Standard |

| | |
|---|---|
| **Document Version** | 2.2.0 |
| **Document Status** | Final |
| **Part of Release** | 4.0 |
| **Revision** | 3 |

**Document Change History**

| Date | Version | Changed by | Change Description |
|---|---|---|---|
| 02.11.2011 | 2.2.0 | AUTOSAR Administration | • Minor bug fixes and editorial changes<br>• Added configurable JIT-update |
| 26.10.2010 | 2.1.0 | AUTOSAR Administration | • Updated: tables for mandatory and optional interfaces, IPDUM020, IPDUM027, IPDUM028, IPDUM032, IPDUM060, IPDUM068, IPDUM083, IPDUM104, IPDUM112_Conf, IPDUM117_Conf, IPDUM143 and IPDUM162<br>• Removed: IPDUM013, IPDUM030, IPDUM050_Conf, IPDUM051_Conf, IPDUM063, IPDUM064, IPDUM065, IPDUM072, IPDUM099 and IPDUM154<br>• Added: pre-compile configuration vari-ant (Chapter 10), IPDUM162_Conf, IPDUM163_Conf, IPDUM164_Conf and IPDUM165 |
| 30.11.2009 | 2.0.0 | AUTOSAR Administration | • Harmonization of FIBEX multiplexing and AUTOSAR multiplexing<br>• Many small corrections based on con-formance tests and validation activities<br>• 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<br>• Defined maximum I-PDU size for FlexRay to 254 bytes<br>• Document meta information extended<br>• 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<br>• Sequence diagrams clarified<br>• Superfluous text removed<br>• Maximum IPDU size clarified<br>• Signature for IpduM_Transmit made consistent with rest of stack.<br>• “Advice for users” revised<br>• Revision Information” added<br>• 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 piece of the SDU of the multiplexed PDU. It is used to distinguish the contents of the multiplexed PDUs from each other.

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

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

# 2 Acronyms and abbreviations

| ***Abbreviation / Acronym:*** | ***Description:*** |
|---|---|
| IpduM | I-PDU Multiplexer |
| dynamic part | see [6] |
| static part | see [6] |
| selector field | see [6] |
| signal | see [7] |
| signal group | see [7] |
| segment | The static or dynamic part may consist of more than one piece. These pieces are called segments. 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 of an I-PDU | IpduM I-PDU with one specific layout and content |

# 3 Related documentation

## 3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[4] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[5] Specification of BSW Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

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

[7] Specification of Communication
AUTOSAR_SWS_COM.pdf

[8] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[9] Concept of Debugging in BSW & RTE Features
AUTOSAR_RS_BSWAndRTEFeatures

## 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 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.⌋(BSW00429)

## 5.2 BSW Scheduler

The BSW-Scheduler (see [5]) schedules the main function of the IpduM.

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

## 5.3 PDU-Router

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

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

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

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

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

## 5.4 COM

The configuration of the IpduM module relies on a corresponding configuration of the AUTOSAR COM module. For each multiplexed I-PDU, there needs to be different I-PDUs configured in the COM module for the static part and each layout of the dynamic part. For further information, see Chapter 7.1 and especially Figure 2.

The IpduM further assumes that the correct selector field values are already contained in the COM's modules I-PDU representing the dynamic parts. See also IPDUM098.

## 5.5 File structure

### 5.5.1 Code file structure

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.⌋()

### 5.5.2 Header file structure

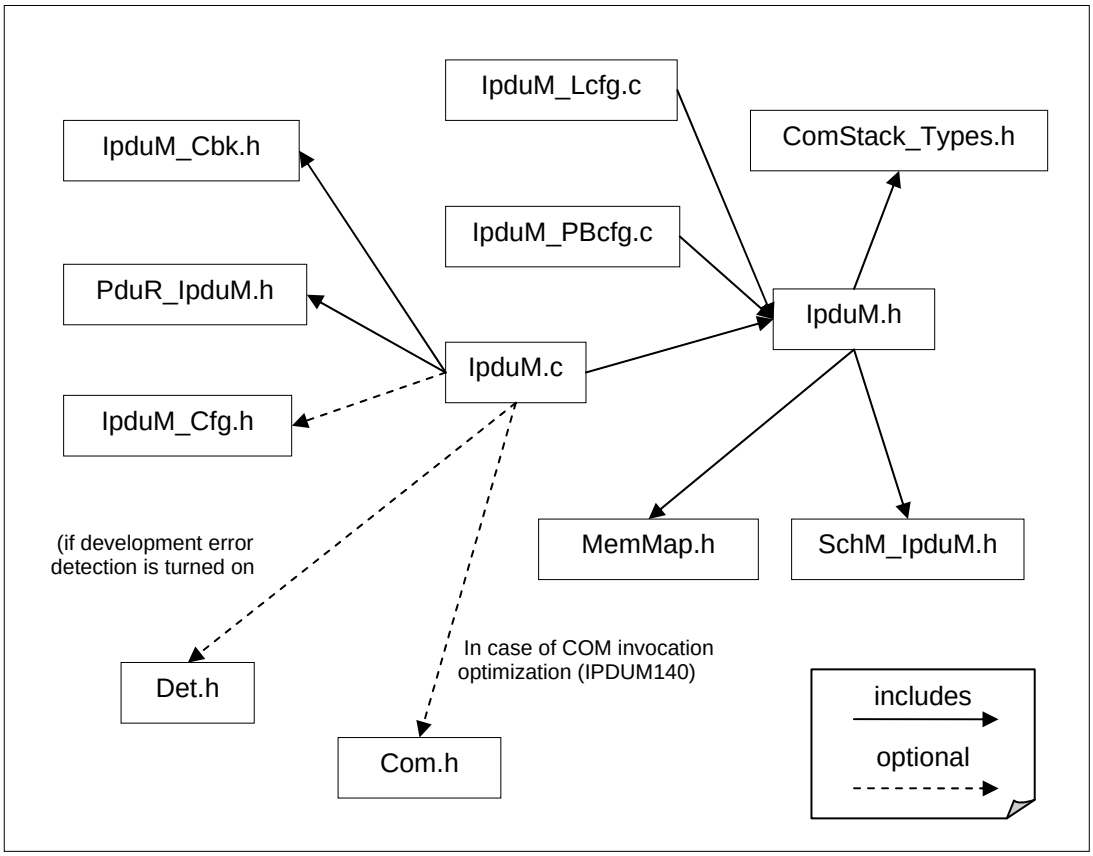


Figure 1 Header File Structure

**[IPDUM148]** ⌈The file IpduM.c shall include IpduM.h, IpduM_Cbk.h, PduR_IpduM.h, and optionally IpduM_Cfg.h, Det.h and Com.h. ⌋(BSW00415)

**[IPDUM149]** ⌈The file IpduM_Lcfg.c shall include IpduM.h. ⌋(BSW00415)

**[IPDUM150]** ⌈The file IpduM_PBcfg.c shall include IpduM.h. ⌋(BSW00415)

**[IPDUM151]** ⌈File IpduM.h shall include MemMap.h, SchM_IpduM.h and ComStack_Types.h.⌋(BSW00415)

**[IPDUM165]** ⌈The IpduM module shall perform Inter Module Checks to avoid integration of incompatible files. The imported include files shall be checked by preprocessing directives. The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

<MODULENAME> is the module's short name of the other (external) module, which provides header files included by the IpduM module.

If the values are not identical to the expected values, an error shall be reported. ⌋(BSW004)

### 5.5.3 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. ⌋(BSW007)

**[IPDUM074]** ⌈The code of the IpduM module shall avoid direct use of compiler and platform specific keywords. ⌋(BSW161)

**[IPDUM075]** ⌈The code of the IpduM module shall indicate all global data with read-only purposes by explicitly assigning the const keyword. ⌋(BSW00309)

**[IPDUM076]** ⌈The IpduM module can use macros instead of functions where source code is used and runtime is critical. ⌋(BSW00330)

**[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. ⌋(BSW00309)

**[IPDUM078]** ⌈The source code of the IpduM module shall not be processor and compiler dependent. ⌋(BSW161)

# 6 Requirements traceability

Document: AUTOSAR requirements on Basic Software [3]

| Requirement | Satisfied by |
|---|---|
| - | IPDUM104 |
| - | IPDUM144 |
| - | IPDUM168 |
| - | IPDUM146 |
| - | IPDUM095 |
| - | IPDUM096 |
| - | IPDUM105 |
| - | IPDUM145 |
| - | IPDUM169 |
| - | IPDUM147 |
| BSW003 | IPDUM037 |
| BSW00309 | IPDUM077, IPDUM075 |
| BSW00314 | IPDUM999 |
| BSW00323 | IPDUM028 |
| BSW00325 | IPDUM999 |
| BSW00326 | IPDUM999 |
| BSW00330 | IPDUM085, IPDUM076 |
| BSW00336 | IPDUM999 |
| BSW00337 | IPDUM106 |
| BSW00338 | IPDUM028, IPDUM027 |
| BSW00339 | IPDUM999 |
| BSW00344 | IPDUM032 |
| BSW00350 | IPDUM027 |
| BSW00357 | IPDUM102 |
| BSW00369 | IPDUM060, IPDUM043, IPDUM044, IPDUM037, IPDUM032 |
| BSW00369;BSW02817 | IPDUM040 |
| BSW00375 | IPDUM999 |
| BSW00377 | IPDUM999 |
| BSW00386 | IPDUM999 |
| BSW004 | IPDUM165, IPDUM039, IPDUM038 |
| BSW00405 | IPDUM032 |
| BSW00406 | IPDUM084, IPDUM083 |
| BSW00407 | IPDUM037 |
| BSW00411 | IPDUM039 |
| BSW00415 | IPDUM151, IPDUM150, IPDUM149, IPDUM148 |

| | |
|---|---|
| BSW00417 | IPDUM999 |
| BSW00422 | IPDUM999 |
| BSW00423 | IPDUM999 |
| BSW00425 | IPDUM103 |
| BSW00427 | IPDUM999 |
| BSW00429 | IPDUM107 |
| BSW00431 | IPDUM999 |
| BSW00432 | IPDUM999 |
| BSW00433 | IPDUM999 |
| BSW00434 | IPDUM999 |
| BSW00437 | IPDUM999 |
| BSW00438 | IPDUM159 |
| BSW005 | IPDUM999 |
| BSW007 | IPDUM073 |
| BSW02800 | IPDUM007, IPDUM004 |
| BSW02801 | IPDUM009 |
| BSW02802 | IPDUM005 |
| BSW02804 | IPDUM006 |
| BSW02806 | IPDUM010 |
| BSW02807 | IPDUM097 |
| BSW02808 | IPDUM004 |
| BSW02809 | IPDUM067, IPDUM068, IPDUM098, IPDUM143 |
| BSW02810 | IPDUM091, IPDUM090, IPDUM089 |
| BSW02811 | IPDUM021 |
| BSW02812 | IPDUM086, IPDUM041, IPDUM042, IPDUM140 |
| BSW02813 | IPDUM022, IPDUM101 |
| BSW02814 | IPDUM019, IPDUM020, IPDUM024, IPDUM023, IPDUM152, IPDUM088, IPDUM087 |
| BSW02816 | IPDUM017, IPDUM015 |
| BSW02818 | IPDUM022 |
| BSW02819 | IPDUM020, IPDUM023 |
| BSW101 | IPDUM032, IPDUM033 |
| BSW161 | IPDUM078, IPDUM074 |
| BSW162 | IPDUM999 |
| BSW164 | IPDUM999 |
| BSW168 | IPDUM999 |
| BSW171 | IPDUM999 |

| ***Requirement*** | ***Satisfied by*** |
|---|---|
| [BSW00344] Reference to link-time configuration | Chapter 10.2.2, IPDUM032 |
| [BSW00404] | Chapter 10.2 |

| | |
|---|---|
| Reference to post build time configuration | |
| [BSW00405] Reference to multiple configuration sets | IPDUM032 |
| [BSW00345] Pre-compile-time configuration | Chapter 10.2.2, IPDUM059_CONF, IPDUM047_CONF, IPDUM048_CONF, IPDUM049_CONF, IPDUM050_CONFIPDUM052_CONF, IPDUM053_CONF, IPDUM056_CONFIPDUM156 |
| [BSW159] Tool-based configuration | not scope of this specification<br>Refers to Configuration WP. |
| [BSW167] Static configuration check-ing | not scope of this specification<br>Refers to Configuration WP. |
| [BSW171] Configurability of optional functionality | not applicable<br>(there is no optional functionality) |
| [BSW170] Data for reconfiguration of AUTOSAR SW-Components | not scope of this specification<br>Refers to Configuration WP. |
| [BSW00380] Separate C-Files for configuration parameters | IPDUM095, IPDUM096<br>implementation specific |
| [BSW00419] Separate C-Files for pre-compile time configuration parameters | Chapter 5.5<br>implementation specific |
| [BSW00381] Separate configuration header file for pre-compile time parameters | Chapter 5.5<br>implementation specific |
| [BSW00412] Separate H-File for configuration parameters | Chapter 5.5<br>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_CONF, IPDUM071_CONF, IPDUM082_CONF, IPDUM130_CONF |
| [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] | Chapter 10.2 |

| | |
|---|---|
| Parameters shall have a range | |
| [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, IPDUM159 |
| [BSW00402] Published information | IPDUM141_CONF, IPDUM142_CONF, IPDUM160 |
| [BSW00375] Notification of wake-up reason | not applicable<br>(this layer cannot perform a wake-up) |
| [BSW101] Initialization interface | IPDUM032, IPDUM033 IPDUM034IPDUM064 IPDUM065 |
| [BSW00416] Sequence of Initialization | not scope of this specification<br>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<br>(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<br>(this module has no connection to the RTE) |
| [BSW00424] BSW main processing function task allocation | not scope of this specification<br>Implementation specific |
| [BSW00425] Trigger conditions for schedulable objects | IPDUM103, IPDUM131_CONF |
| [BSW00426] Exclusive areas in BSW modules | not scope of this specification<br>Implementation specific |
| [BSW00427] ISR description for BSW modules | not applicable<br>(module does not provide ISRs) |
| [BSW00428] | Chapter 8.6 |

| | |
|---|---|
| Execution order dependencies of main processing functions | |
| [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, IPDUM153 |
| [BSW00338] Detection and Reporting of development errors | IPDUM027, IPDUM028, IPDUM059_CONF, IPDUM132_CONF, IPDUM154 |
| [BSW00369] Do not return development error codes via API | IPDUM032, IPDUM037, IPDUM040, IPDUM043, IPDUM044, IPDUM060 |
| [BSW00339] Reporting of production relevant errors and exceptions | not applicable (module does not define any production relevant errors) |
| [BSW00422] Pre—debouncing 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 | IPDUM038, IPDUM039, IPDUM059_CONF, IPDUM134_CONF, IPDUM165 |
| [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]<br>No hard coded horizontal interfaces within MCAL | not applicable<br>(not scope of this specification) |
| [BSW00415]<br>User dependent include files | IPDUM148, IPDUM149, IPDUM150, IPDUM151 |
| [BSW164]<br>Implementation of interrupt service routines | not applicable<br>(module does not provide ISRs) |
| [BSW00325]<br>Runtime of interrupt service routines | not applicable<br>(module does not provide ISRs) |
| [BSW00326]<br>Transition from ISRs to OS tasks | not applicable<br>(module does not provide ISRs) |
| [BSW00342]<br>Usage of source code and object code | Chapter 10.2 |
| [BSW00343]<br>Specification and configuration of time | Chapter 10.2 |
| [BSW160]<br>Human-readable configuration data | Chapter 10.2 |
| [BSW007]<br>HIS MISRA C | IPDUM073 |
| [BSW00300]<br>Module naming convention | Figure 1 |
| [BSW00413]<br>Accessing instances of BSW modules | not scope of this specification<br>implementation specific |
| [BSW00347]<br>Naming separation of different instances of BSW drivers | not scope of this specification<br>implementation specific |
| [BSW00305]<br>Self-defined data types naming convention | Chapter 8.3.1 |
| [BSW00307]<br>Global variables naming convention | not scope of this specification<br>implementation specific |
| [BSW00310]<br>API naming convention | Chapter 8.4 and 8.5 |
| [BSW00373]<br>Main processing function naming convention | Chapter 8.6 |
| [BSW00327]<br>Error values naming convention | IPDUM026 |
| [BSW00335]<br>Status values naming convention | not scope of this specification<br>implementation specific |
| [BSW00350]<br>Development error detection keyword | IPDUM027 |
| [BSW00408]<br>Configuration parameter naming convention | Chapter 10.2 |
| [BSW00410] | not scope of this specification |

| | |
|---|---|
| Compiler switches shall have defined values | 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 |
| [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]<br>Definition of global data | not scope of this specification<br>implementation specific |
| [BSW00309]<br>Global data with read-only constraint | IPDUM075, IPDUM077 |
| [BSW00371]<br>Do not pass function pointers via API | Chapter 8.4 and 8.5 |
| [BSW00358]<br>Return type of init functions | Chapter 8.4.1 |
| [BSW00414]<br>Parameter of init function | Chapter 8.4.1 |
| [BSW00376]<br>Return type and parameters of main processing functions | Chapter 8.6 |
| [BSW00359]<br>Return type of callback functions | Chapter 8.5 |
| [BSW00360]<br>Parameters of callback functions | Chapter 8.5 |
| [BSW00329]<br>Avoidance of generic interfaces | Chapter 8 |
| [BSW00330]<br>Usage of macros / inline functions instead of functions | IPDUM076, IPDUM085 |
| [BSW00331]<br>Separation of error and status values | Chapter 8 |
| [BSW009]<br>Module User Documentation | not scope of this specification<br>implementation specific |
| [BSW00401]<br>Documentation of multiple instances of configuration parameters | Chapter 10.2 |
| [BSW172]<br>Compatibility and documentation of scheduling strategy | not scope of this specification<br>implementation specific |
| [BSW010]<br>Memory resource documentation | not scope of this specification<br>implementation specific |
| [BSW00333]<br>Documentation of callback function context | not scope of this specification<br>implementation specific |
| [BSW00374]<br>Module vendor identification | Chapter 10.3 |
| [BSW00379]<br>Module identification | Chapter 10.3 |
| [BSW003]<br>Version identification | IPDUM037, IPDUM059_CONF |
| [BSW00318]<br>Format of module version numbers | Chapter 10.3 |

| | |
|---|---|
| [BSW00321]<br>Enumeration of module version numbers | not scope of this specification<br>implementation specific |
| [BSW00341]<br>Microcontroller compatibility documentation | not scope of this specification<br>implementation specific |
| [BSW00334]<br>Provision of XML file | not scope of this specification<br>Refers to Configuration WP |

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

| ***Requirement*** | ***Satisfied by*** |
|---|---|
| [BSW02800]<br>Exactly one selector field per PDU | IPDUM004, IPDUM007 |
| [BSW02801]<br>Size of the selector field | IPDUM009, IPDUM052_CONF |
| [BSW02802]<br>Position of the selector field | IPDUM005, IPDUM155 |
| [BSW02815]<br>Compile Time configuration of the selector field | IPDUM052_CONF |
| [BSW02803]<br>Unused values of the selector field | IPDUM011 |
| [BSW02804]<br>Support for static and dynamic parts of the PDU | IPDUM006 |
| [BSW02808]<br>Support of multiplexed PDUs with a static part of length “zero” | IPDUM004, IPDUM133_CONF |
| [BSW02809]<br>Initialization of multiplexed PDUs | IPDUM068, IPDUM067, IPDUM098, IPDUM143 |
| [BSW02806]<br>Semantic of the multiplexer | IPDUM010 |
| [BSW02810]<br>Routing of multiplexed PDUs on sender side | IPDUM063IPDUM089, IPDUM090, IPDUM091, IPDUM112_CONF |
| [BSW02816]<br>Combining of multiplexed PDUs on sender side | IPDUM015, IPDUM017, IPDUM114_CONF, IPDUM120_CONF, IPDUM121_CONF, IPDUM123IPDUM125_CONF, IPDUM126_CONF, IPDUM127_CONF, IPDUM128_CONF, IPDUM129_CONF, IPDUM157_CONF |
| [BSW02811]<br>Triggering condition on sender side | IPDUM021, IPDUM052_CONF |
| [BSW02812]<br>Routing of multiplexed PDUs on receiver side | IPDUM041, IPDUM042, IPDUM086, IPDUM108_CONF, IPDUM109_CONF, IPDUM140 |
| [BSW02817]<br>De-multiplexing PDUs on receiver side | IPDUM040, IPDUM113_CONF, IPDUM114_CONF, IPDUM115_CONF |
| [BSW02813]<br>Routing of Send Confirmations | IPDUM022, IPDUM050_CONFIPDUM101 |
| [BSW02818]<br>Confirmation replication of multiplexed PDUs | IPDUM022, IPDUM124_CONF, IPDUM163_Conf, IPDUM164_Conf , IPDUM158_ConfIPDUM050_CONF |
| [BSW02814]<br>Correct confirmation handling of multiplexed PDUs | IPDUM023, IPDUM024, IPDUM019, IPDUM020, IPDUM087,IPDUM088, IPDUM152 |
| [BSW02807]<br>No Runtime Overhead for systems without PDU multiplexing | IPDUM097 |
| [BSW02819] | IPDUM020, IPDUM023 |

| *Requirement* | *Satisfied by* |
|---|---|
| No queuing of transmission requests on sender side | |

## AUTOSAR Release 4.0 Concept Incorporation

| *Concept* | *Satisfied by* |
|---|---|
| Debugging concept [9] | IPDUM144, IPDUM145, IPDUM146, IPDUM147 |

# 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. ⌋(BSW02800, BSW02808)

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. ⌋(BSW02802)

**[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 segments. ⌋(BSW02804)

**[IPDUM007]** ⌈There shall be only one selector field within one multiplexed I-PDU. ⌋(BSW02800)

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

**[IPDUM009]** ⌈The selector field of one I-PDU shall have a configurable size between one and eight contiguous bits. ⌋(BSW02801)

**[IPDUM010]** ⌈The position of the selector field within the I-PDU shall be defined by configuration. ⌋(BSW02806)

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

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

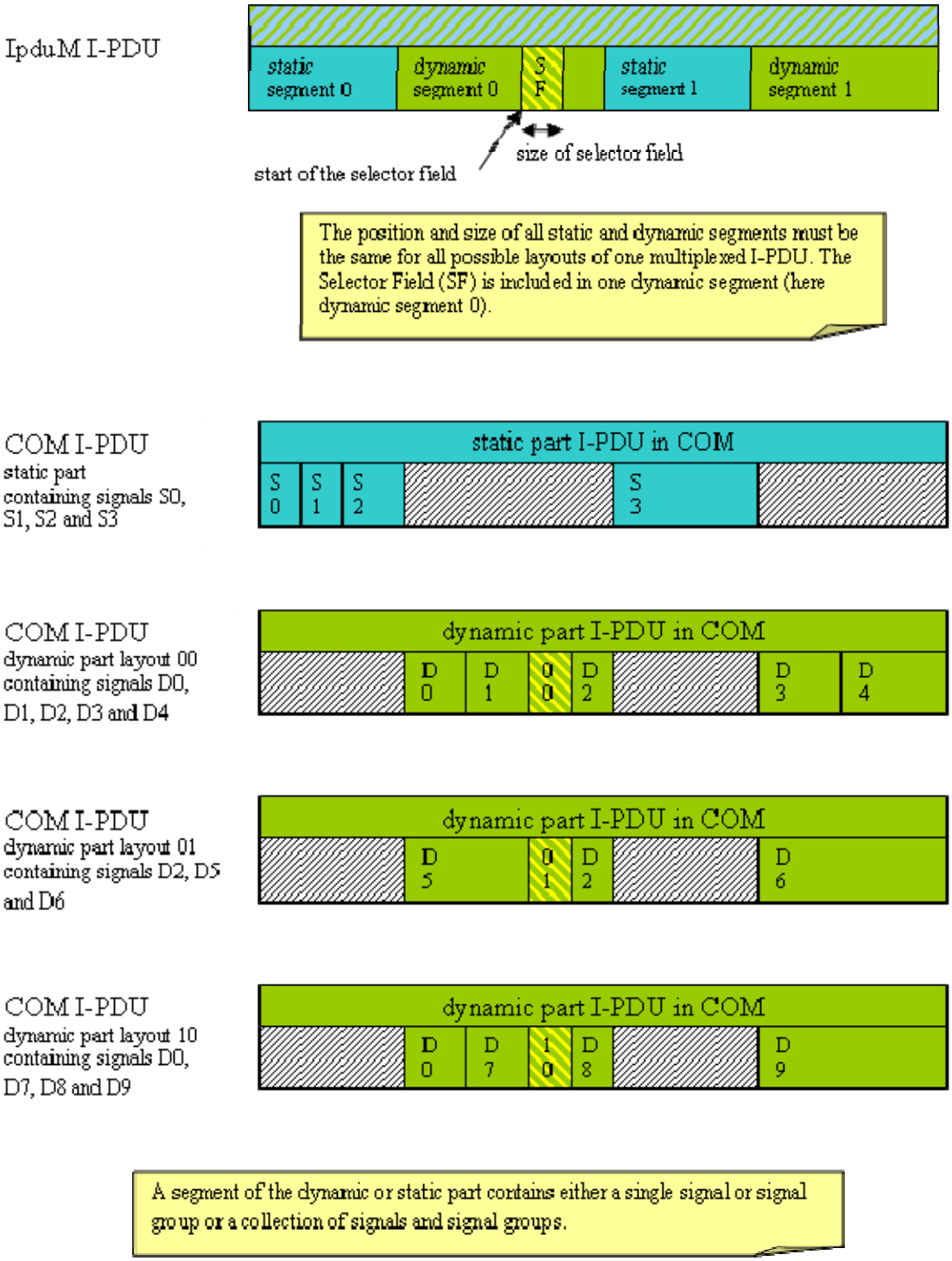


**Figure 2 Possible layout of a multiplexed I-PDU**

## 7.2 Overview

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

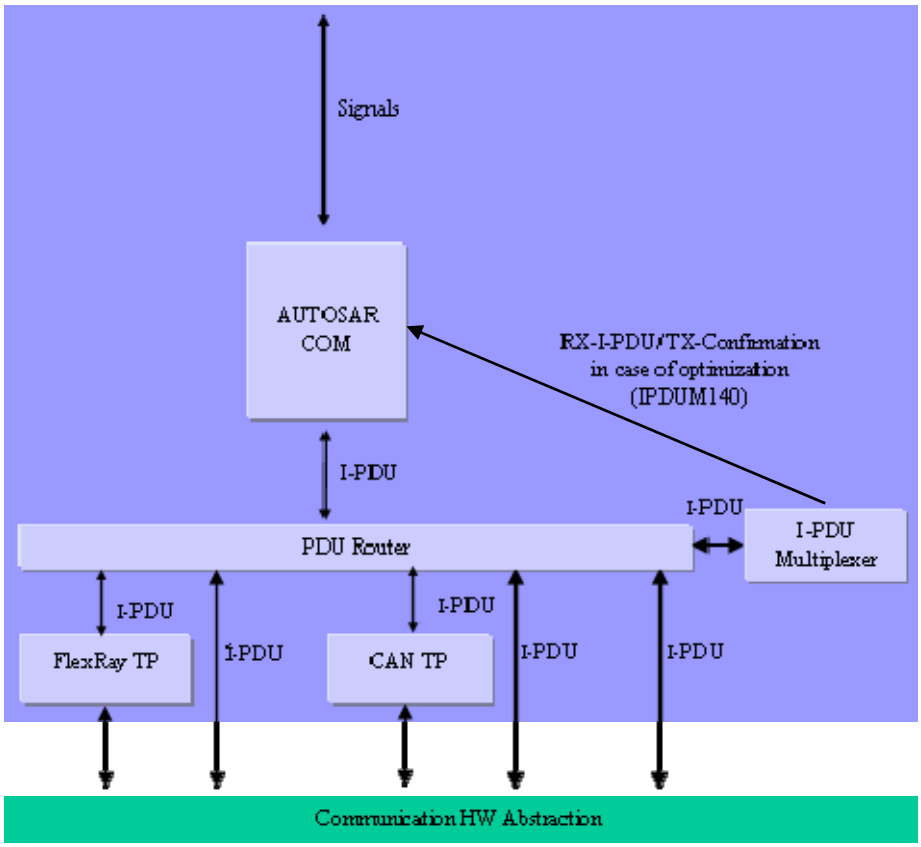


**Figure 3 I-PDU Multiplexer in the AUTOSAR Architecture**

**[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. ⌋(BSW02807)

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

**[IPDUM098]** ⌈The IpduM module shall not set the selector field. ⌋(BSW02809)

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

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

**[IPDUM140]** ⌈It shall be allowed to optimize the RX- and Tx-Confirmation path from the IpduM module via the PDU-Router module to the COM layer to call the COM API directly from the IpduM module without including the PDU-Router. This shall be indicated by setting the published parameter IpduMRxDirectComInvocation to TRUE, see IPDUM142_CONF. ⌋(BSW02812)

In case of the COM invocation, optimization as defined above IpduM.c needs to include Com.h, see Figure 1 Header File Structure.

## 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.

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

IpduM_Init uses the PduR_IpduMTriggerTransmit function to retrieve the initial I-PDU values from the COM module. Therefore, the COM module needs to be initialized via Com_Init before the IpduM module can be initialized via IpduM_Init. The integrator must take care of this dependency.

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

For the I-PDU data transmission pathway through the IpduM module, a buffer is allocated inside the IpduM module. This buffer needs to be initialized in case it is transmitted before it has been fully populated with data by COM. The initialization data for this buffer is derived by using configuration data from the IpduMTxRequest container as follows:

a. **[IPDUM067]** ⌈The IpduM internal buffer shall first be filled with the pattern defined in the configuration parameter IpduMIPduUnusedAreasDefault. ⌋(BSW02809)

b. **[IPDUM068]** ⌈The initial signal values for the initial dynamic part, referenced by IpduMinitialDynamicPart, shall be fetched from the COM module via PduR_IpduMTriggerTransmit and the configured IpduMSegment operations of that dynamic part shall be processed. ⌋(BSW02809)

c. **[IPDUM143]** ⌈The initial signal values for the static part shall be fetched from the COM module via PduR_IpduMTriggerTransmit and the configured IpduM-

Segment operations of the static part shall be processed. ⌋(BSW02809)

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

## 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.

**[IPDUM015]** ⌈For a multiplexed I-PDU IpduM shall merge the corresponding two COM I-PDUs representing the associated static part and the last received dynamic part into one single IpduM I-PDU with a new unique I-PDU ID. IpduM shall send out this new IpduM I-PDU to the PDU-Router module, see also Figure 2. ⌋(BSW02816)

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

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

### 7.4.1 Transmission request

The IpduM module provides an IpduM_Transmit function so that the PDU-R is able to initiate the transmission of an I-PDU; see 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/ modes as defined in IPDUM021 and IPDUM125_CONF. ⌋(BSW02816)

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. ⌋(BSW02814)

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

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

**[IPDUM020]** ⌈In case the IpduMTxConfirmationTimeout was configured to a value greater than 0, as long as the corresponding timeout timer has not elapsed, and no transmission confirmation for that multiplexed I-PDU was received, the function IpduM_Transmit shall not allow a new transmission request from the upper layer with a COM I-PDU that belongs to the same IpduM I-PDUs.⌋(BSW02814, BSW02819)

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

**[IPDUM152]** ⌈As long as the timeout (defined in the configuration parameter IpduMTxConfirmationTimeout) has not elapsed and as long as no transmission confirmation for the IpduM I-PDU is received, the function IpduM_Transmit shall return with E_NOT_OK for a new transmission request from the upper layer with a COM I-PDU that belongs to the same IpduM I-PDUs.⌋(BSW02814)

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

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

It 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 the following trigger conditions/ modes:

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

For configuration, see IPDUM052_CONF.」(BSW02811)

The four trigger conditions/ modes defined by IPDUM021 allow controlling the transmission mode of the new assembled I-PDU by the transmission modes of the single I-PDUs sent by COM, see also [7].

Not all of four trigger conditions/ modes defined by IPDUM021 allow guaranteeing the minimum delay time between consecutive transmissions of different instances of multiplexed I-PDUs, because if the transmission is triggered by static and dynamic part or only by the dynamic part, COM does not take care for the minimum delay time. COM treats the static part and the different dynamic parts as unrelated standalone I-PDUs.

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

### 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.

**[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.⌋(BSW02813, BSW02818)

Depending on the configuration of IpduMTxDynamicConfirmation (IPDUM163_Conf)

and IpduMTxStaticConfirmation (IPDUM164_Conf), the IpduM will pass zero, one or two confirmations towards COM for one send request. The number of confirmations given to the upper layer does not depend on the IpduMTxTriggerMode.

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

**[IPDUM023]** ⌈If the Tx-Confirmation is not received within the configured timeout IpduMTxConfirmationTimeout the IpduM shall allow new transmission requests for this specific I-PDU after timeout is elapsed.⌋(BSW02814, BSW02819)

**[IPDUM024]** ⌈The IpduM shall discard unexpected Tx-Confirmations silently. This may happen if a previously requested transmit request has been timed out, but is confirmed now.⌋(BSW02814)

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

## 7.5 Reception

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

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

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

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

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

## 7.6 Error classification

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

| | ***Type or error*** | ***Relevance*** | ***Related error code*** | ***Value [hex]*** |
|---|---|---|---|---|
| **IPDUM026:** | API service called with wrong parameter | Development | IPDUM_E_PARAM | 10 |
| **IPDUM162:** | API service called with a NULL pointer. In case of this error, the API service shall return immediately without any further action, except for reporting this development error. | Development | IPDUM_E_PARAM_POINTER | 11 |
| **IPDUM153:** | API service used without module initialization | Development | IPDUM_E_UNINIT | 20 |

**[IPDUM106]**「Development error values are of type uint8.」(BSW00337)

## 7.7 Error detection and notification

The detection of development errors can be configured at pre-compile time via the configuration parameter IpduMDevErrorDetect (IPDUM132_Conf).

**[IPDUM027]**「If IpduMDevErrorDetect is configured to FALSE, the IpduM module shall not report any development errors.」(BSW00338, BSW00350)

**[IPDUM028]**「If IpduMDevErrorDetect is configured to TRUE, all IpduM APIs shall check their input parameters and report detected errors to DET by IPDUM_E_PARAM for normal parameter and IPDUM_E_PARAM_POINTER for pointer parameters.」(BSW00338, BSW00323)

## 7.8 Debugging

**[IPDUM144]**「Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.」()

**[IPDUM145]**「All type definitions of variables, which shall be debugged, shall be accessible by the header file IpduM.h.」()

**[IPDUM146]**「The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-“sizeof”.」()

**[IPDUM147]**「Variables available for debugging shall be described in the respective Basic Software Module Description.」()

# 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 |

⌋ (BSW00357)

## 8.2 Type definitions

### 8.2.1 IpduM_ConfigType

**[IPDUM159]** ⌈

| | |
|---|---|
| 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. |

⌋ (BSW00438)

## 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(`<br>`  const IpduM_ConfigType* config`<br>`)` |
| 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. |

」(BSW00344, BSW00405, BSW101, BSW00369)

**[IPDUM033]**「The function IpduM_Init shall initialize all module-related global variables. 」(BSW101)

**[IPDUM083]**「In case, the configuration parameter IpduMDevErrorDetect equals TRUE: if the parameter config does not reference a valid configuration, the function IpduM_Init shall raise the development error IPDUM_E_PARAM_POINTER. 」(BSW00406)

**[IPDUM084]**「The behavior of the IpduM is unspecified until a correct call to IpduM_Init is made. 」(BSW00406)

### 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: | 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. |

」(BSW00407, BSW00369, BSW003)

**[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). 」(BSW004)

**[IPDUM039]**「The function IpduM_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: IpduMVersionInfoApi. 」(BSW004, BSW00411)

**[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.」(BSW00330)

### 8.3.3 IpduM_Transmit

**[IPDUM043]**「

| | | |
|---|---|---|
| Service name: | IpduM_Transmit | |
| Syntax: | `Std_ReturnType IpduM_Transmit(`<br>`  PduIdType PdumTxPduId,`<br>`  const PduInfoType* PduInfoPtr`<br>`)` | |
| 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.<br>Range: 0..(maximum number of I-PDU IDs which are multiplexed) - 1 |
| | PduInfoPtr | A pointer to a structure with I-PDU related data that shall be transmitted: data length and pointer to I-SDU buffer |
| Parameters (inout): | None | |
| Parameters (out): | None | |
| Return value: | Std_ReturnType | E_OK: Transmit request is accepted<br>E_NOT_OK: Transmit request is not accepted |
| Description: | Service is called by the PDU-Router to request a transmission. | |

」(BSW00369)

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

」(BSW00369; BSW02817)

**[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.」(BSW02812)

**[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.」(BSW02812)

**[IPDUM086]**「The function IpduM_RxIndication shall be callable in interrupt context, e.g. from receive interrupt.」(BSW02812)

### 8.4.2 IpduM_TxConfirmation

**[IPDUM044]**「

| | | |
|---|---|---|
| Service name: | IpduM_TxConfirmation | |
| Syntax: | `void IpduM_TxConfirmation(`<br>`  PduIdType TxPduId`<br>`)` | |
| Service ID[hex]: | 0x40 | |
| Sync/Async: | Synchronous | |
| Reentrancy: | Reentrant for different PduIds. Non reentrant for the same PduId. | |
| Parameters (in): | TxPduId | ID of the I-PDU that has been transmitted. |
| Parameters (inout): | None | |
| Parameters (out): | None | |
| Return value: | None | |
| Description: | The lower layer communication module confirms the transmission of an I-PDU. | |

」(BSW00369)

**[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.」(BSW02814)

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 a transmit interrupt.」(BSW02814)

### 8.4.3 IpduM_TriggerTransmit

**[IPDUM060]**「

| | | |
|---|---|---|
| Service name: | IpduM_TriggerTransmit | |
| Syntax: | `Std_ReturnType IpduM_TriggerTransmit( PduIdType TxPduId, PduInfoType* PduInfoPtr )` | |
| Service ID[hex]: | 0x41 | |
| Sync/Async: | Synchronous | |
| Reentrancy: | Reentrant for different PduIds. Non reentrant for the same PduId. | |
| Parameters (in): | TxPduId | ID of the SDU that is requested to be transmitted. |
| | PduInfoPtr | Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength. |
| Parameters (inout): | None | |
| Parameters (out): | None | |
| Return value: | Std_ReturnType | E_OK: SDU has been copied and SduLength indicates the number of copied bytes.<br>E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data. |
| Description: | The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module. | |

」(BSW00369)

**[IPDUM090]**「The function IpduM_TriggerTransmit shall copy the contents of its I-PDU transmit buffer to the I-PDU buffer given by PduInfoPtr. 」(BSW02810)

**[IPDUM091]**「The IpduM shall take care about the data consistency during providing the data. 」(BSW02810)

**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. 」(BSW02810)

## 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( void )` |
| 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. |

」(BSW00425)

**[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. 」(BSW02813)

## 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 |
|---|---|

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

### 8.6.2 Optional Interfaces

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

**[IPDUM105]** ⌈

| API function | Description |
|---|---|
| Det_ReportError | Service to report development errors. |
| PduR_IpduMTransmit | Requests transmission of an I-PDU. |
| PduR_IpduMRxIndication | Indication of a received I-PDU from a lower layer communication module. |
| PduR_IpduMTriggerTransmit | The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module. |
| PduR_IpduMTxConfirmation | The lower layer communication module confirms the transmission of an I-PDU. |

⌋ ()

### 8.6.3 Configurable interfaces

Not applicable

# 9 Sequence diagrams

## 9.1 Transmission of a multiplexed I-PDU and Transmit confirmation

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

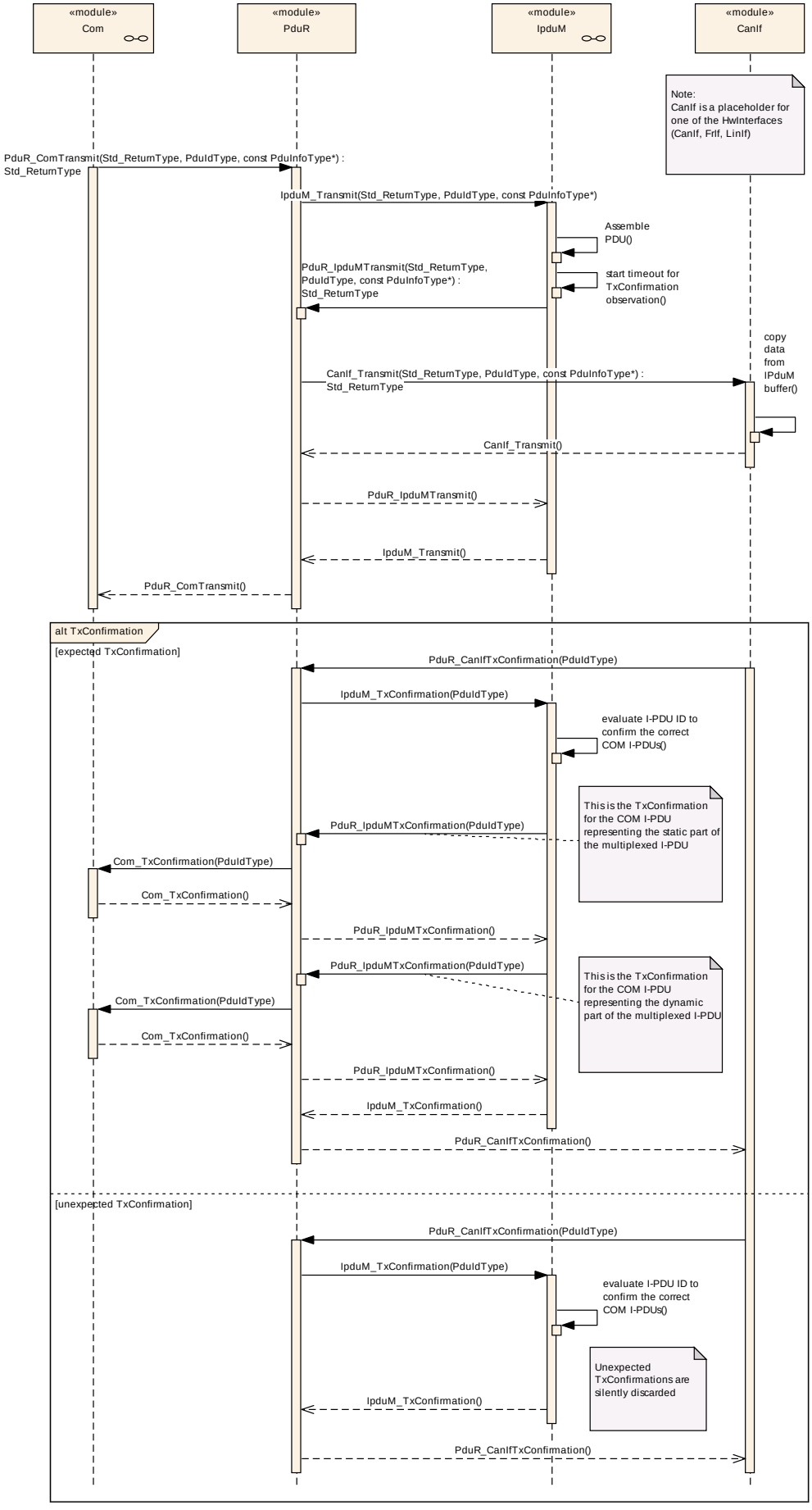
«module»
Com
«module»
PduR
«module»
IpduM
«module»
CanIf
Note:
CanIf is a placeholder for one of the HwInterfaces (CanIf, Frlf, LinIf)
PduR_ComTransmit(Std_ReturnType, PduIdType, const PduInfoType*) : Std_ReturnType
IpduM_Transmit(Std_ReturnType, PduIdType, const PduInfoType*)
Assemble PDU()
PduR_IpduMTransmit(Std_ReturnType,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

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

## 9.2 Transmission of a multiplexed I-PDU without Trigger

The following sequence chart shows a transmit request initiated by the COM layer. Because of the configuration of the IpduM, no transmit request for the IpduM I-PDU takes place. For configuration see IPDUM052_CONF.

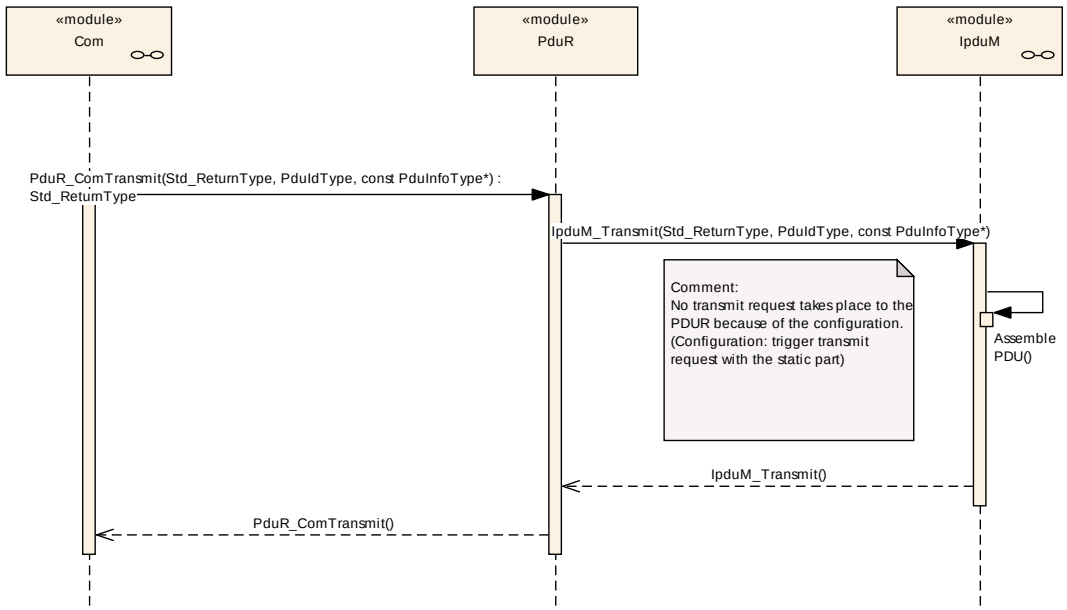


**Figure 5 Transmission of a multiplexed I-PDU without triggering**

## 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.

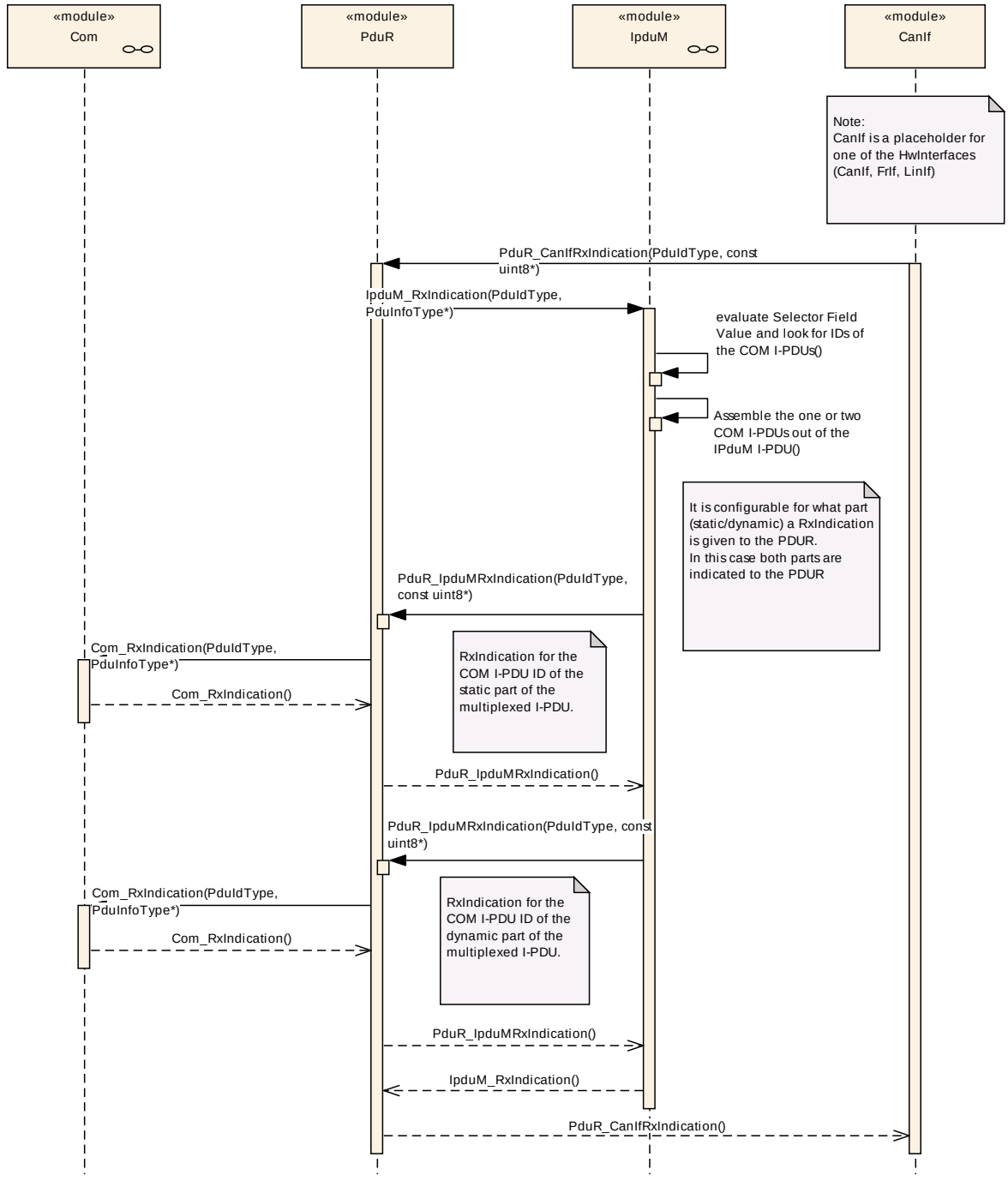


**Figure 6 Reception of a multiplexed I-PDU**

## 9.4 Trigger Transmit

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

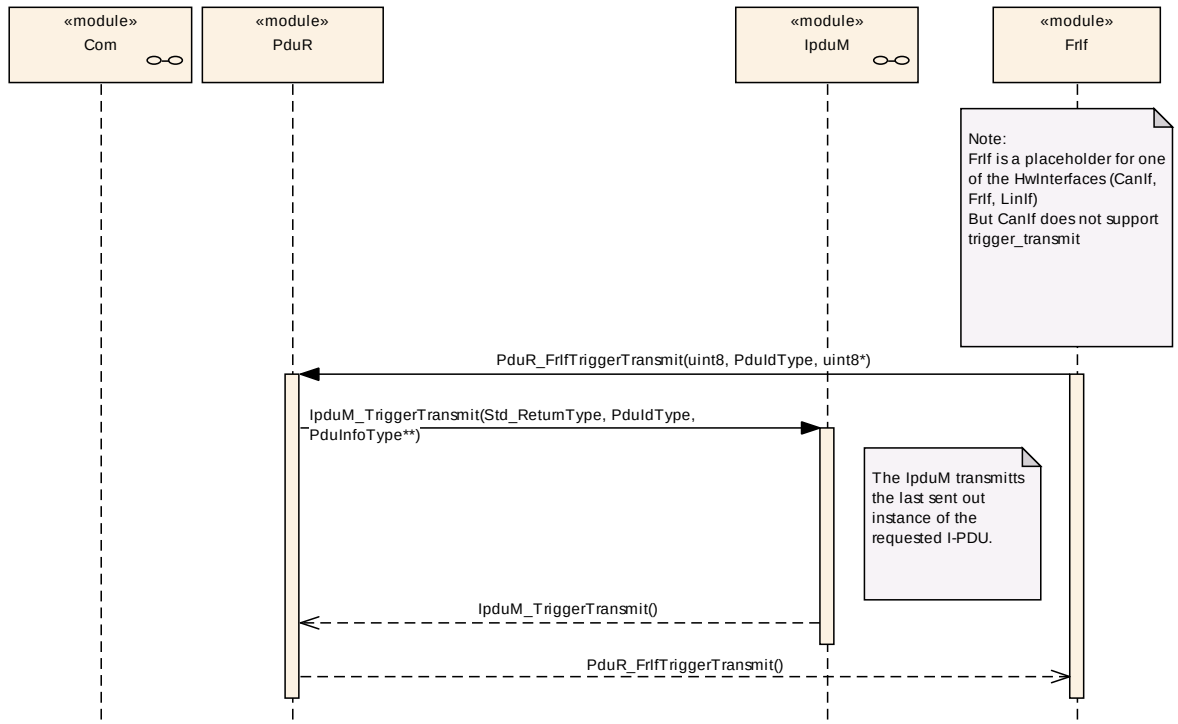


**Figure 7 Trigger Transmit request from interface layer**

## 9.5 Missing Transmit Confirmation

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

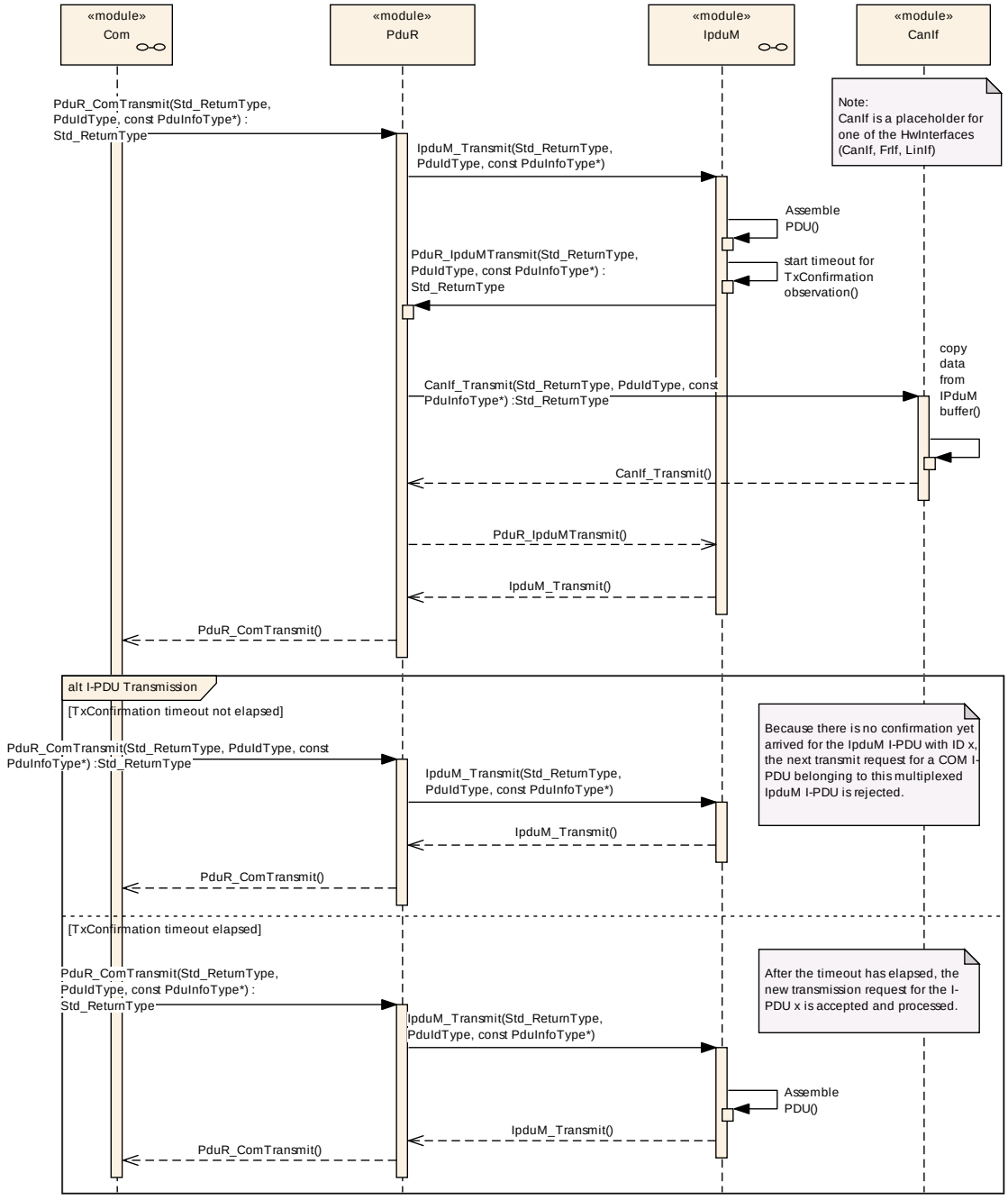


**Figure 8 Missing Transmit Confirmation**

# 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 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

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

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

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

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

### 10.2.2 Configuration overview

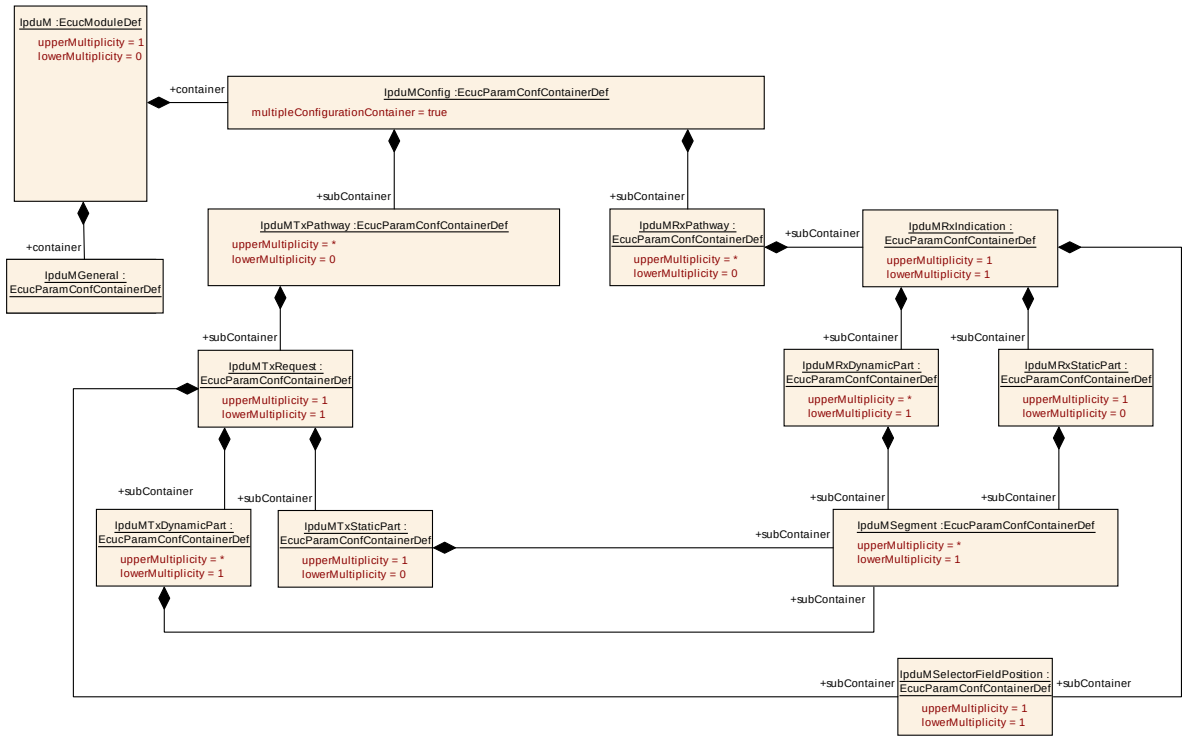


**Figure 9 IpduM Configuration Overview**

### 10.2.3 IpduM

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

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

### 10.2.4 IpduMGeneral

| | |
|---|---|
| SWS Item | **IPDUM130_Conf :** |
| Container Name | IpduMGeneral |
| Description | Contains the general configuration parameters of IpduM. |
| Configuration Parameters | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM131_Conf :** | | |
| Name | IpduMConfigurationTimeBase | | |
| Description | The cycle time with which IpduM_MainFunction should be invoked (in seconds). | | |
| Multiplicity | 1 | | |
| Type | EcucFloatParamDef | | |
| Range | 0 .. 3600 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM132_Conf :** | | |
| Name | IpduMDevErrorDetect | | |
| Description | Active/Deactivate the detection of development errors, for production code this parameter has to be False. True: error detection activated False: error detection deactivated | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM133_Conf :** | | |
|---|---|---|---|
| Name | IpduMStaticPartExists | | |
| Description | This is to allow optimizations in the case the IpduM will never be used with a static part. Note that this is a pre-compile option. If this is set to False then it will not be possible to add static parts after compilation. True: A static part may exist. False: A static part will never exist. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM134_Conf :** | | |
|---|---|---|---|
| Name | IpduMVersionInfoApi | | |
| Description | Active/Deactivate the version information API. true: version information activated false: version information deactivated | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| No Included Containers |
|---|

### 10.2.5 IpduMTxPathway

| SWS Item | **IPDUM070_Conf :** |
|---|---|
| Container Name | IpduMTxPathway |
| Description | Contains the configuration parameters transmitted I-PDUs by the IpduM module. |
| Configuration Parameters | |

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

### 10.2.6 IpduMTxRequest

| SWS Item | **IPDUM052_Conf :** |
|---|---|
| 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_Conf :** | | |
|---|---|---|---|
| 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 | EcucEnumerationParamDef | | |
| Range | BIG_ENDIAN | -- | |
| | LITTLE_ENDIAN | -- | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | | | |

| SWS Item | **IPDUM121_Conf :** | | |
|---|---|---|---|
| Name | IpduMIPduUnusedAreasDefault | | |
| Description | IpduM module fills not used areas of an I-PDU with this bit-pattern If this attribute is omitted the IpduM module does not fill the I-PDU. | | |
| Multiplicity | 0..1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 0 .. 255 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM158_Conf :** | | |
|---|---|---|---|
| Name | IpduMTxConfirmationPduId | | |
| Description | The handle Id to be used by the PduR to confirm the transmission of this Pdu. The existence of this parameter is essential for the PduR generation tool to actually find a symbolicNameValue for the OutgoingPdu. | | |
| Multiplicity | 0..1 | | |
| Type | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| Range | 0 .. 65535 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM124_Conf :** |
|---|---|
| Name | IpduMTxConfirmationTimeout |

| | | | |
|---|---|---|---|
| Description | This timeout (in seconds) defines the timeout period for monitoring the reception of the TxConfirmation. It is not used when an I-PDU is requested using the trigger transmit API. | | |
| Multiplicity | 0..1 | | |
| Type | EcucFloatParamDef | | |
| Range | 0 .. 3600 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM125_Conf :** | | |
| Name | IpduMTxTriggerMode | | |
| Description | Selects whether to send the multiplexed I-PDU immediately or at some later date. | | |
| Multiplicity | 1 | | |
| Type | EcucEnumerationParamDef | | |
| Range | DYNAMIC_PART_TRIGGER | | Writing the I-PDU representing the dynamic part does trigger a sending of the I-PDU. |
| | NONE | | Only the buffer in the IpduM are written but not send is triggered, used for IpduM I-PDUs which are requested by TriggerTransmit. |
| | STATIC_OR_DYNAMIC_PART_TRIGGER | | Writing the I-PDU representing the static or the dynamic part does trigger a sending of the I-PDU. |
| | STATIC_PART_TRIGGER | | Writing the I-PDU representing the static part does trigger a sending of the I-PDU. |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | |
|---|---|
| SWS Item | **IPDUM157_Conf :** |
| Name | IpduMInitialDynamicPart |
| Description | Reference to the dynamic part that shall be used to initialize this multiplexed TX-I-PDU. |
| Multiplicity | 1 |

| | | | |
|---|---|---|---|
| Type | Reference to [ IpduMTxDynamicPart ] | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM120_Conf :** | | |
| 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 |
| IpduMSelectorFieldPositi-on | 1 | Specifies the position of the selector field in the outgoing I-PDU. |
| IpduMTxDynamicPart | 1..* | This (These) included container(s) must exist for each unique selector field value for this outgoing IpduM I-PDU. |
| IpduMTxStaticPart | 0..1 | This included container configures the static part, if present. |

### 10.2.7 IpduMTxDynamicPart

| | |
|---|---|
| SWS Item | **IPDUM056_Conf :** |
| Container Name | IpduMTxDynamicPart |
| Description | Configuration parameters for an instance of a TxRequest call into the IpduM. When a Tx Request with the IpduMTxDynamicHandleId is re-ceived by the IpduM, all segments as defined by this container are copied from the incoming I-PDU into the outgoing I-PDU buffer and then the send mode honoured. 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_Conf :** | | |
| 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 | EcucBooleanParamDef | | |
| Default value | false | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | |
|---|---|
| SWS Item | **IPDUM163_Conf :** |
| Name | IpduMTxDynamicConfirmation |

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

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM127_Conf :** | | |
| Name | IpduMTxDynamicHandleId | | |
| Description | This is an incoming handle id. When the handle of an incoming Tx Request matches this, the bits fields (see IpduMSegment) are copied and the IpduMTxTriggerMode is honored. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| Range | 0 .. 65535 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: External | | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM126_Conf :** | | |
| 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 |
| IpduMSegment | 1..* | This is a list of all segments to be copied from the incoming I-PDU to the outgoing I-PDU. |

### 10.2.8 IpduMTxStaticPart

| | |
|---|---|
| SWS Item | **IPDUM082_Conf :** |
| Container Name | IpduMTxStaticPart |
| Description | Configuration parameters for an instance of a Tx_Request call into the IpduM. When a Tx Request with the IpduMTxStaticHandleId is received by the IpduM, all segments as defined by this container are copied from the incoming I-PDU into the outgoing I-PDU buffer and then the send mode honoured. 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_Conf :** | | |
|---|---|---|---|
| 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 | EcucBooleanParamDef | | |
| Default value | false | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM164_Conf :** | | |
|---|---|---|---|
| Name | IpduMTxStaticConfirmation | | |
| Description | A transmit request can be confirmed by the lower layer. If this parameter is set to true a confirmation of the I-PDU in COM representing the static part is generated. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM129_Conf :** | | |
|---|---|---|---|
| Name | IpduMTxStaticHandleId | | |
| Description | This is an incoming handle id. When the handle of an incoming Tx Request matches this, the segments are copied (IPduMSegment) and the IpduMTxTriggerMode is honored. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| Range | 0 .. 65535 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: External | | |

| SWS Item | **IPDUM128_Conf :** | | |
|---|---|---|---|
| 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 |
|---|---|---|
| IpduMSegment | 1..* | This is a list of all segments to be copied from the incoming I-PDU to the outgoing I-PDU. |

### 10.2.9 IpduMRxPathway

| SWS Item | **IPDUM071_Conf :** |
|---|---|
| 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 |

### 10.2.10 IpduMRxIndication

| SWS Item | **IPDUM047_Conf :** |
|---|---|
| Container Name | IpduMRxIndication |
| Description | Contains the configuration for incoming RxIndication calls. |
| Configuration Parameters | |

| SWS Item | **IPDUM162_Conf :** | | |
|---|---|---|---|
| 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 | EcucEnumerationParamDef | | |
| Range | BIG_ENDIAN | -- | |
| | LITTLE_ENDIAN | -- | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | | | |

| SWS Item | **IPDUM109_Conf :** |
|---|---|
| Name | IpduMRxHandleId |
| Description | This is the I-PDU ID of the incoming I-PDU. If an incoming RxIndication's I-PDU ID matches this value then it is |

| | | | |
|---|---|---|---|
| | unpacked according to the specification in this container. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef (Symbolic Name generated for this parameter) | | |
| Range | 0 .. 65535 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | | | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM108_Conf :** | | |
| 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 |
| IpduMRxDynamicPart | 1..* | 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. |
| IpduMSelectorFieldPosition | 1 | This contains the location of the selector field. At run-time, the selector field is used to select which dynamic part is unpacked. |

### 10.2.11 IpduMRxDynamicPart

| | |
|---|---|
| SWS Item | **IPDUM048_Conf :** |
| 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 IpduM_Selector_Value, 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. |
| Configuration Parameters | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM113_Conf :** | | |
| Name | IpduMRxSelectorValue | | |
| Description | This is the selector value that this container refers to. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 0 .. 255 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |

| | |
|---|---|
| Scope / Dependency | scope: local |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM112_Conf :** | | |
| 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 |
| IpduMSegment | 1..* | The DynamicPart can be separated in multiple segments within the multiplexed PDU. |

## 10.2.12 IpduMRxStaticPart

| | |
|---|---|
| SWS Item | **IPDUM049_Conf :** |
| Container Name | IpduMRxStaticPart |
| Description | This container contains the configuration for the static part of incoming RxIndication calls. On reception, the new outgoing I-PDU for the static part is constructed as defined by the segments of this container and sent out with the I-PDU ID referenced by IpduMOutgoingStaticPduRef. |
| Configuration Parameters | |

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM115_Conf :** | | |
| 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 |
| IpduMSegment | 1..* | The StaticPart can be separated in multiple segments within the multiplexed PDU. |

### 10.2.13 IpduMSegment

| SWS Item | **IPDUM053_Conf :** |
|---|---|
| Container Name | IpduMSegment |
| Description | This contains the location and the length of a segment. A segment must fit inside the I-PDU. The segment in the source I-PDU that is located at the IpduMSegmentPosition is copied to the same position in the destination I-PDU. |
| Configuration Parameters | |

| SWS Item | **IPDUM114_Conf :** | | |
|---|---|---|---|
| Name | IpduMSegmentLength | | |
| Description | Length of the segment in bits. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 1 .. 2032 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| SWS Item | **IPDUM159_Conf :** | | |
|---|---|---|---|
| Name | IpduMSegmentPosition | | |
| Description | Segments bit position in the multiplexed Pdu. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 0 .. 2031 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| No Included Containers |
|---|

### 10.2.14 IpduMSelectorFieldPosition

| SWS Item | **IPDUM054_Conf :** |
|---|---|
| Container Name | IpduMSelectorFieldPosition |
| Description | This contains the location and the length of the selector field. |
| Configuration Parameters | |

| SWS Item | **IPDUM160_Conf :** | | |
|---|---|---|---|
| Name | IpduMSelectorFieldLength | | |
| Description | Length of the selector field in bits. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 1 .. 8 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |

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

| | | | |
|---|---|---|---|
| SWS Item | **IPDUM161_Conf :** | | |
| Name | IpduMSelectorFieldPosition | | |
| Description | Selector field bit position in the multiplexed Pdu. | | |
| Multiplicity | 1 | | |
| Type | EcucIntegerParamDef | | |
| Range | 0 .. 2031 | | |
| Default value | -- | | |
| ConfigurationClass | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | X | VARIANT-LINK-TIME |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| |
|---|
| No Included Containers |

## 10.2.15 IpduMConfig

| | |
|---|---|
| SWS Item | **IPDUM059_Conf :** |
| 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.<br>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 |
| IpduMRx-Pathway | 0..* | includes information about received I-PDUs |
| IpduMTxPathway | 0..* | includes information about sent I-PDUs |

## 10.3 Published Information

**[IPDUM170]** 「The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1]. 」()

Additional module-specific published parameters are listed below if applicable.

### 10.3.1 IpduMPublishedInformation

| SWS Item | **IPDUM141_Conf :** |
|---|---|
| Container Name | IpduMPublishedInformation |
| Description | Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, since they are published information. |
| Configuration Parameters | |

| SWS Item | **IPDUM142_Conf :** | | |
|---|---|---|---|
| Name | IpduMRxDirectComInvocation | | |
| Description | If set to TRUE the COM invocation optimization as defined in IPDUM140 is implemented. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| ConfigurationClass | Published Information | X | All Variants |
| Scope / Dependency | | | |

| No Included Containers |
|---|

## 10.4 Configuration Rules

### 10.4.1 Selector Field

**[IPDUM155]** 「The selector fields shall not cross any byte-boundary within the I-PDU. 」(BSW02802)

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

**[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.」(BSW02803)

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

### 10.4.2 Byte Order

The byte order of all signals and the selector field of a multiplexed I-PDU is restricted to be the same, see IPDUM162_Conf. Any necessary byte order conversion shall be handled within the COM module. The multiplexed I-PDUs in COM and IpduM have to be configured consistently to have the same endianness.

**[IPDUM166]**「The endianness of signals of the de-multiplexed I-PDUs configured in COM must match the endianness of the corresponding multiplexed I-PDU in IpduM as configured per IpduMByteOrder (IPDUM162_Conf).」()

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

# 11 Changes to Release 3.0

## 11.1 Deleted SWS Items

| *SWS Item* | *Rationale* |
|---|---|
| IPDUM008 | obsolete requirement |
| IPDUM013 | requirement to other module |
| IPDUM029 | obsolete requirement |
| IPDUM030 | redundant to reformulated requirement IPDUM028 |
| IPDUM034<br>IPDUM035 | configuration process of initial values changed |
| IPDUM050_Conf | simplified confirmation configuration |
| IPDUM051_Conf | simplified confirmation configuration |
| IPDUM063 | requirement to other module |
| IPDUM064 | requirement was implementation specific |
| IPDUM065 | requirement was implementation specific |
| IPDUM069 | became obsolete while updating IpduM initialization process |
| IPDUM072 | requirement to other module |
| IPDUM092 | turned into a note |
| IPDUM099 | requirement to other module |
| IPDUM117_Conf | simplified confirmation configuration |
| IPDUM118_Conf | simplified confirmation configuration |
| IPDUM119_Conf | simplified confirmation configuration |
| IPDUM123 | was removed from the MetaModel against the ECU Configuration Parameter XML File; the length of the I-PDU has to looked up via PduLength parameter of referenced by the IpduMOutgoingPduRef |
| IPDUM154 | redundant to reformulated requirement IPDUM027 |

## 11.2 Replaced SWS Items

| *SWS Item of Release 1* | *replaced by SWS Item* | *Rationale* |
|---|---|---|
| IPDUM002 | IPDUM148<br>IPDUM149<br>IPDUM150<br>IPDUM151 | CT SWS Analysis required to separate requirements regarding include file structure |
| IPDUM122 | IPDUM157_CONF | configuration process of initial values changed |

## 11.3 Changed SWS Items

| *SWS Item* | *Rationale* |
|---|---|
| IPUDM006 | rephrased due to FIBEX harmonization |
| IPDUM015 | rephrased requirement and added reference to Figure |
| IPDUM017 | clarified term trigger conditions |
| IPDUM020 | clarified the case when no timeout was configured |
| IPDUM021 | added a reference to clarify TriggerTransmit case |
| IPDUM027 | clarified development error reporting |
| IPDUM028 | clarified development error reporting |
| IPDUM032 | added const to configuration pointer |
| IPDUM052_Conf | update description of included container IpduMBitField |

| | |
|---|---|
| IPDUM060 | harmonized trigger transmit APIs within the communication stack |
| IPDUM067 | clarified term buffer |
| IPDUM068 | updated IpduM initialization process |
| IPDUM083 | added development error IPDUM_E_PARAM_POINTER |
| IPDUM102 | updated to actual BSW UML model |
| IPDUM104 | updated table of mandatory interfaces |
| IPDUM105 | updated table of optional interfaces |
| IPDUM112_Conf | clarified description of PDU reference |
| Chapter 10 | the configuration structure was updated in order to harmonize the IpduM configuration with FIBEX and the new configuration variant VARIANT-PRE-COMPILE has been added |

## 11.4 Added SWS Items

| ***SWS Item*** | ***Rationale*** |
|---|---|
| IPDUM140 | turned note about direct COM invocation optimization into a requirement |
| IPDUM141_CONF | added new configuration container IpduMPublishedInformation |
| IPDUM142_CONF | added new published parameter IpduMRxDirectComInvocation |
| IPDUM143 | updated IpduM initialization process |
| IPDUM144 | added according to the debugging concept |
| IPDUM145 | added according to the debugging concept |
| IPDUM146 | added according to the debugging concept |
| IPDUM147 | added according to the debugging concept |
| IPDUM148-IPDUM151 | see replaced SWS items |
| IPDUM152 | split IPDUM020 to IPDUM020 and IPDUM152 |
| IPDUM153 | split IPDUM026 to IPDUM026 and IPDUM153 |
| IPDUM154 | split IPDUM027 to IPDUM027 and IPDUM154 |
| IPDUM155 | restricted selector-field to be within one byte |
| IPDUM158_Conf | added IpduMTxConfirmationPduId |
| IPDUM159 | added requirement ID for IpduM_ConfigType |
| IPDUM160 | added requirement ID for published information |
| IPDUM161_Conf | explicit configuration of selector field |
| IPDUM001_PI | rework of Published Information |
| IPDUM162 | added development error IPDUM_E_PARAM_POINTER |
| IPDUM162_Conf | added configuration parameter IpduMByteOrder |
| IPDUM163_Conf | simplified confirmation configuration |
| IPDUM164_Conf | simplified confirmation configuration |
| IPDUM165 | added requirement for explicit version checking of include file |
| IPDUM166 | added configuration restriction to clarify that the IpduM module shall provide no endianness conversion mechanism |

# 12 Not applicable requirements

**[IPDUM999]**「These requirements are not applicable to this specification.」
(BSW171, BSW00375, BSW00437, BSW168, BSW00423, BSW00427, BSW00431, BSW00432, BSW00433, BSW00434, BSW00336, BSW00339, BSW00422, BSW00417, BSW00386, BSW162, BSW005, BSW164, BSW00325, BSW00326, BSW00314, BSW00377)