Document Title	Specification of PDU Router
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	035
Document Classification	Standard

Document Version	2.2.3
Document Status	Final
Part of Release	3.1
Revision	0001

Document Change History

Date	Version	Changed by	Change Description
15.08.2008	2.2.3	AUTOSAR Administration	Layout adaptations
23.06.2008	2.2.2	AUTOSAR Administration	Legal disclaimer revised
22.01.2008	2.2.1	AUTOSAR Administration	• Correction figure 3
13.11.2007	2.2.0	AUTOSAR Administration	• Variants have been renamed. • New Callbacks PduR_LinTpChangeParameterConfirmation, PduR_FrTpChangeParameterConfirmation • PduR_CanTpChangeParameterConfirmation has been added. • New API's PduR_ChangeParameterRequest, PduR_CancelTransmitRequest has been added • New Typedefines PduR_ParameterValueType, PduR_CancelReasonType has been added • Document meta information extended • Small layout adaptations made
31.01.2007	2.1.0	AUTOSAR Administration	• Clarifications added (FIFO, TxConf, ...) • Unnecessary development errors removed • SchM_PduR.h and MemMap.h added • Corrections of configuration parameters • More details in Chapter 11 • Legal disclaimer revised • Release Notes added • "Advice for users" revised • "Revision Information" added

28.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template. • Major changes in chapter 10 • Structure of document changed partly • Other changes see chapter
20.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

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 and API for the AUTOSAR PDU Router module.

The PDU Router module provides services for routing of I-PDUs (Interaction Layer Protocol Data Units) between the following modules:

- communication interface modules (e.g. LINIF, CANIF, and FlexRayIf)
- Transport Protocol modules (e.g. CAN TP, FlexRay TP)
- AUTOSAR Diagnostic Communication Manager (DCM) and Transport Protocol modules (e.g. CAN TP, FlexRay TP)
- AUTOSAR COM and communication interface modules (e.g. LINIF, CANIF, or FlexRayIf) or I-PDU Multiplexer
- I-PDU Multiplexer and communication interface modules (e.g. LINIF, CANIF, or FlexRayIf)

PDUs are identified by static PDU IDs. The PDU Router module determines the destination of a PDU by using the PDU ID and a static configuration table. I-PDUs are used for the data exchange of the modules directly above the PDU Router, e.g. AUTOSAR COM and AUTOSAR DCM. The routing operation of the PDU Router module does not modify the I-PDU, it simply forwards the I-PDU to the destination module. In case of TP routing, forwarding of the I-PDU is started before the full I-PDU is received (“routing on-the-fly”).

The PDU Router module provides an API for modules below the PDU Router module (communication interface modules and transport protocol modules) and an API for modules directly above (e.g. DCM and COM) [1]. Furthermore the PDU Router module provides an interface for the I-PDU multiplexer (IPDUM) which is located beside the PDU Router. All these interfaces are constructed such that the operations required to pass data between the lower and upper layers are minimized.

The PDU Router module provides 1:n routing for single frame communication; i.e. (a) I-PDUs to be sent or received via interface modules and (b) I-PDUs to be sent or received within a single frame via TP modules. For Network Management data exchange the PDU Router module is bypassed. Figure 1 gives an overview of the AUTOSAR communication structure.

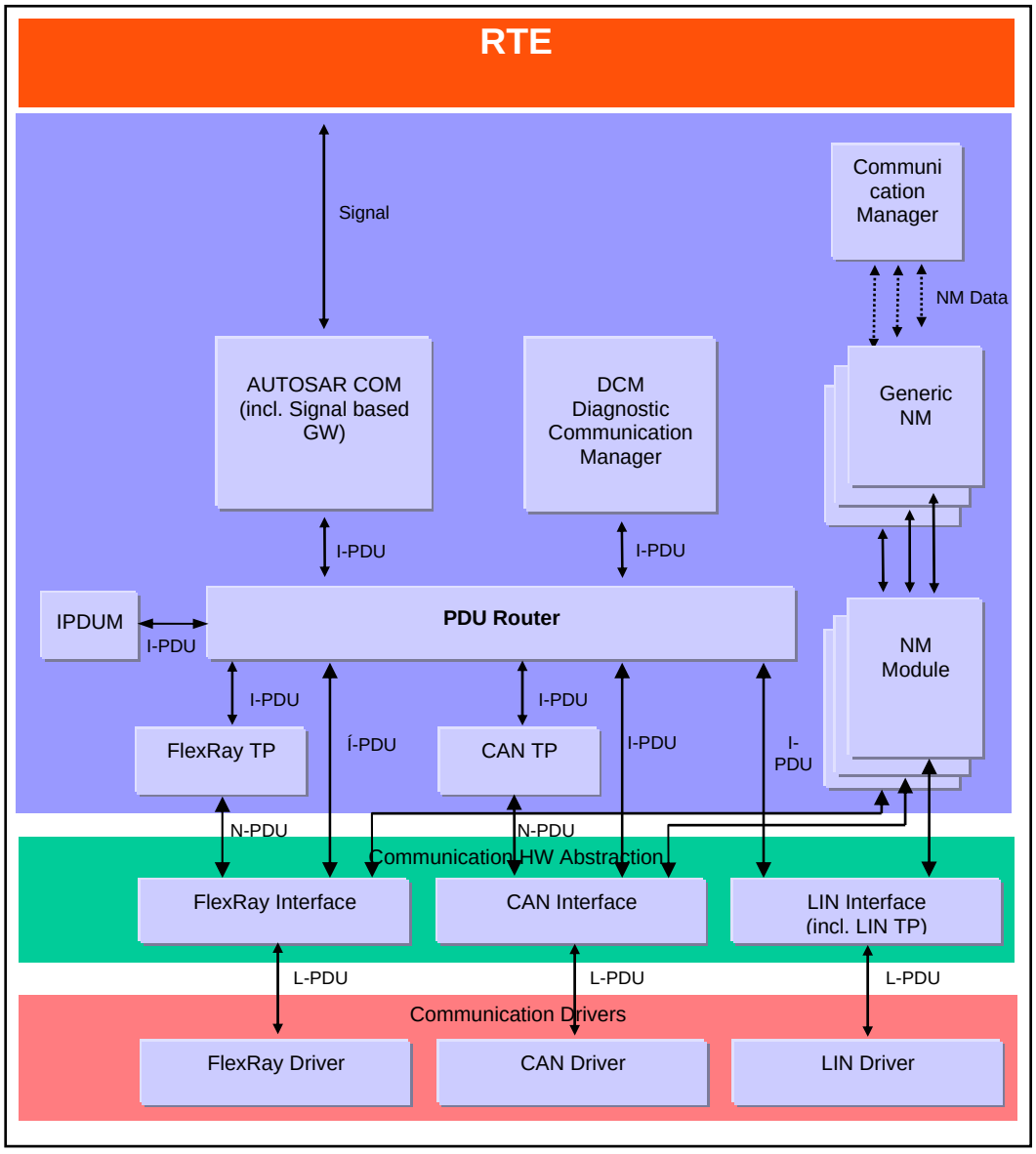

Figure 1: Communication Structure

The PDU Router module is part of the AUTOSAR Basic SW, and is mandatory instantiated in every AUTOSAR ECU.

The detailed PDU Router module structure is shown in Figure 2. It mainly consists of two parts:

- The **PDU Router routing tables**: static routing tables describing the routing attributes for each PDU to be routed. The routing tables can be updated post-build time in the programming state of the ECU (see section 7.5).

- The **PDU Router Engine**: the actual code performing routing actions according to the PDU Router routing tables. The router engine has to deal with two translations:

- The **PDU Router UP Translation (PRUPT)**: Translation of PDU IDs and API of the PDU Router module to the related module above the PDU Router module (e.g.: COM, DCM, …) or the IPDUM.

- The **PDU Router LO Translation (PRLOT)**: Translation of PDU IDs and API of the PDU Router module to the related module below the PDU Router module (FlexRay Interface, CAN Interface, FlexRay TP, …) or the IPDUM.

Additionally the PDU Router Engine provides a minimum routing capability to be able to route specific PDUs without using the PDU Router routing tables. Thus access to the DCM for the activation of the ECU bootloader may be supported even when the post-build time configurable PDU Router routing tables are corrupted. The minimum routing settings are separated from the PDU Router routing tables and cannot be changed after build-time.

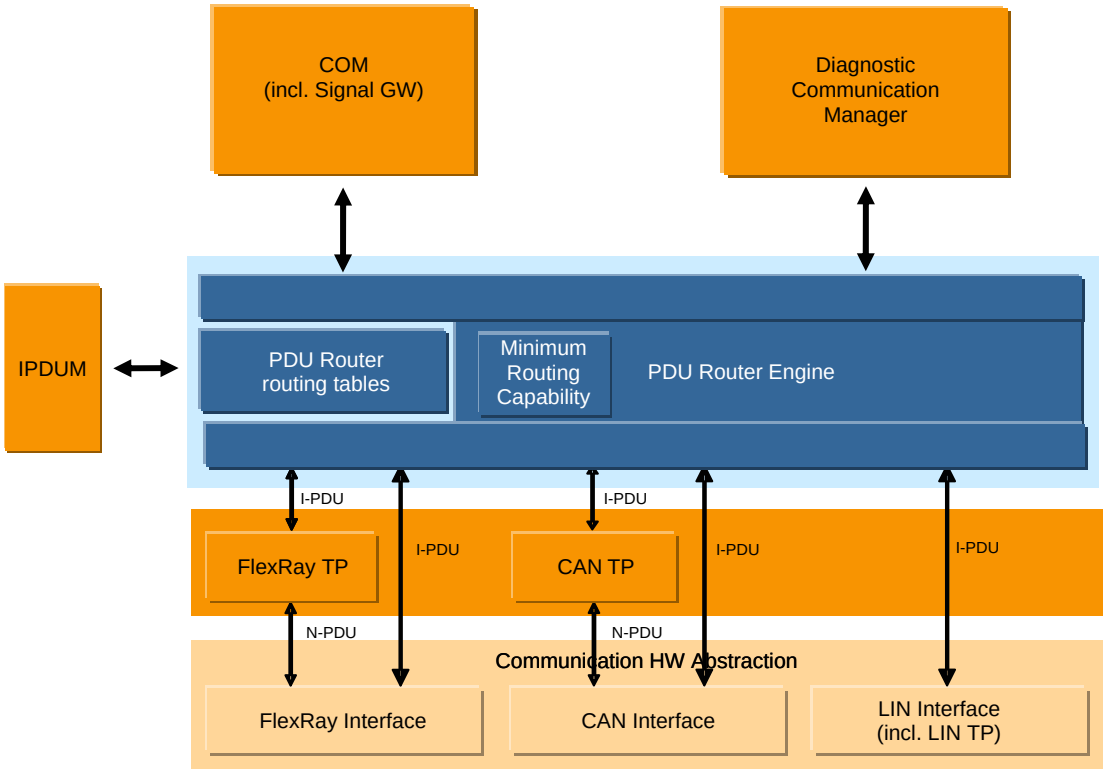

Figure 2: Detailed PDU Router Structure

2 Acronyms and abbreviations

The following acronyms and abbreviations have a local scope and are therefore not contained in the AUTOSAR glossary.

Acronym:	***Description:***
Upper Layer Modules (Up)	Modules above the PDU Router. Currently this layer includes COM and Diagnostic Communication Manager (DCM).
Lower Layer Modules (Lo)	Modules below the PDU Router. Currently this layer includes CAN, LIN, FlexRay communication interface modules and the respective TP modules.
PDU Router	Module that transfers I-PDUs from one module to another module. The PDU Router module can be utilized for gateway operations and for internal routing purposes.
routing-on-the-fly	Gateway capability; routing between two communication modules where forwarding of data is started before all data have been received.
multicast operation	Simultaneous transmission of PDUs to a group of receivers.
data provision	Provision of data to interface modules. (a) direct data provision: data to be transmitted are provided directly at the transmit request (b) trigger transmit data provision: data to be transmitted are not provided at the transmit request, but will be retrieved by the interface module via a callback function

Abbreviation:	***Description:***
<Up>	An instance of an upper layer module
<Lo>	An instance of a lower layer module
PDU ID	PDU Identifier

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf,

[2] Requirements on Gateway,
AUTOSAR_SRS_Gateway.pdf

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

[4] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[5] Specification of Communication Stack Types
AUTOSAR_SWS_ComStackTypes.pdf

[6] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[7] Specification of CAN Interface
AUTOSAR_SWS_CAN_Interface.pdf

[8] Specification of CAN Transport Layer
AUTOSAR_SWS_CAN_TP.pdf

[9] Specification of LIN Interface
AUTOSAR_SWS_LIN_Interface.pdf

[10] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRay_Interface.pdf

[11] Specification of FlexRay Transport Layer
AUTOSAR_SWS_FlexRay_TP.pdf

[12] Specification of Communication
AUTOSAR_SWS_COM.pdf

[13] Specification of DCM
AUTOSAR_SWS_DCM.pdf

[14] Specification of DEM
AUTOSAR_SWS_DEM.pdf

[15] Specification of ECU Configuration
AUTOSAR_ECU_Configuration.pdf

[16] Specification of ECU ConfigurationParameters
AUTOSAR_ECU_ConfigurationParameters.pdf

[17] Specification of I-PDU Multiplexer
AUTOSAR_SWS_IPDUM.pdf

[18] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSWMDTemplate.pdf

3.2 Related standards and norms

[19] LIN Communication Protocol, LIN specification package, Revision 2.0, September 23, 2003

[20] CAN Communication Protocol, ISO11898 – Road vehicles - Controller area network (CAN)

[21] ISO 15765-2(2003-11-11), Road vehicles – Diagnostics on Controller Area Networks (CAN) – Part2: Network layer services

[22] FlexRay Communication Protocol, FlexRay Communication Systems Protocol Specification Version 2.1

4 Constraints and assumptions

4.1 Limitations

1. The PDU Router module does not provide mechanisms for signal extraction or conversion.

2. The PDU Router module does not provide mechanisms for data integrity checking (like checksums).

3. The PDU Router module does not change or modify the I-PDU.

4. The PDU Router module does not make any PDU payload dependent routing decisions.

5. The PDU Router module does not support routing between TP modules and communication interface modules or vice versa.

6. The PDU Router module does not support 1:n routing of I-PDUs which are sent or received via a TP module and require multiple frames for transmission.

7. The PDU Router module itself does not support routing of I-PDUs between communication interface modules with rate conversion. (This functionality will be supported in cooperation with an upper layer module, e.g. COM as shown in section 9.4, Figure 15).

4.2 Applicability to car domains

In this version the PDU Router module has not been specified to work with the MOST communication network. Thus the applicability to multimedia and telematic car domains may be limited.

5 Dependencies to other modules

The PDU Router module depends on the API and capabilities of the used communication hardware abstraction layer modules and the used communication service layer modules. Basically the API functions required by the PDU Router module are:

- Communication interface modules:
  <Lo>If_Transmit (e.g. CanIf_Transmit, FrIf_Transmit, LinIf_Transmit)

- Transport Protocol Modules:
  <Lo>Tp_Transmit (e.g. CanTp_Transmit, FrTp_Transmit, LinTp_Transmit)

- Upper layer modules which use TP:
  <Up>_ProvideRxBuffer (e.g. Dcm_ProvideRxBuffer),
  <Up>_ProvideTxBuffer (e.g. Dcm_ProvideTxBuffer),
  <Up>_RxIndication (e.g. Dcm_RxIndication)
  <Up>_TxConfirmation (e.g. Dcm_TxConfirmation)

- Upper layer modules which do not use TP:
  <Up>_RxIndication (e.g. Com_RxIndication),
  <Up>_TxConfirmation (e.g. Com_TxConfirmation),
  <Up>_TriggerTransmit (e.g. Com_TriggerTransmit)

- I-PDU Multiplexer:
  Ipdum_Transmit
  Ipdum_TxConfirmation
  Ipdum_TriggerTransmit
  Ipdum_RxIndication

5.1 File structure

5.1.1 Code file structure

PDUR226: The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- PduR_Cfg.c – for pre-compile time configuration parameters implemented as “const”,
- PduR_Lcfg.c – for link time configurable parameters and
- PduR_PBcfg.c – for post build time configurable parameters.

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

5.1.2 Header file structure

PDUR292: General PDU Router module definitions shall be defined in PduR.h.

PDUR293: Type definitions of the PDU Router module shall be defined in PduR_Types.h.

PDUR159: Pre-compile-time configuration data of the PDU Router module shall be defined in PduR_Cfg.h.

PDUR216: Due to the high number of communication modules related to the PDU Router module, the functions used by the different modules shall be declared in separate header files:

- `PduR_Com.h` (8.3.5.1)
- `PduR_Dcm.h` (8.3.6.1)
- `PduR_CanIf.h` (8.3.2.1, 8.3.2.2),
  `PduR_CanTp.h` (8.3.2.3, 8.3.2.4, 8.3.2.5, 8.3.2.6)
- `PduR_FrIf.h` (8.3.3.1, 8.3.3.2, 8.3.3.3),
  `PduR_FrTp.h` (8.3.3.4, 8.3.3.5, 8.3.3.6, 8.3.3.7)
- `PduR_LinIf.h` (8.3.4.1, 8.3.4.2, 8.3.4.3),
  `PduR_LinTp.h` (8.3.4.4, 8.3.4.5, 8.3.4.6, 8.3.4.7)
- `PduR_Ipdum.h` (8.3.7.1)

PDUR132: The include file structure regarding the specifics of the PDU Router module shall be constructed as shown in Figure 3.

- `PduR_Types.h` shall include `ComStack_Types.h`
- `PduR.h` shall include `PduR_Types.h`, `PduR_Cfg.h`
- `PduR_<module>.h` (i.e. `PduR_Com.h`, `PduR_Dcm.h`, `PduR_CanIf.h`, `PduR_CanTp.h`, `PduR_FrIf.h`, `PduR_FrTp.h`, `PduR_LinIf.h`, `PduR_LinTp.h`, `PduR_Ipdum.h`) shall include `PduR.h`
- `PduR.c` shall include `Dem.h`, `SchM_PduR.h`, `MemMap.h` and all `PduR_<module>.h`, `<module>.h/<module>_Cbk.h`, and `Det.h` if the related pre-compile time configuration parameter is enabled (e.g. PduRFrIfSupport for PduR_FrIf.h).

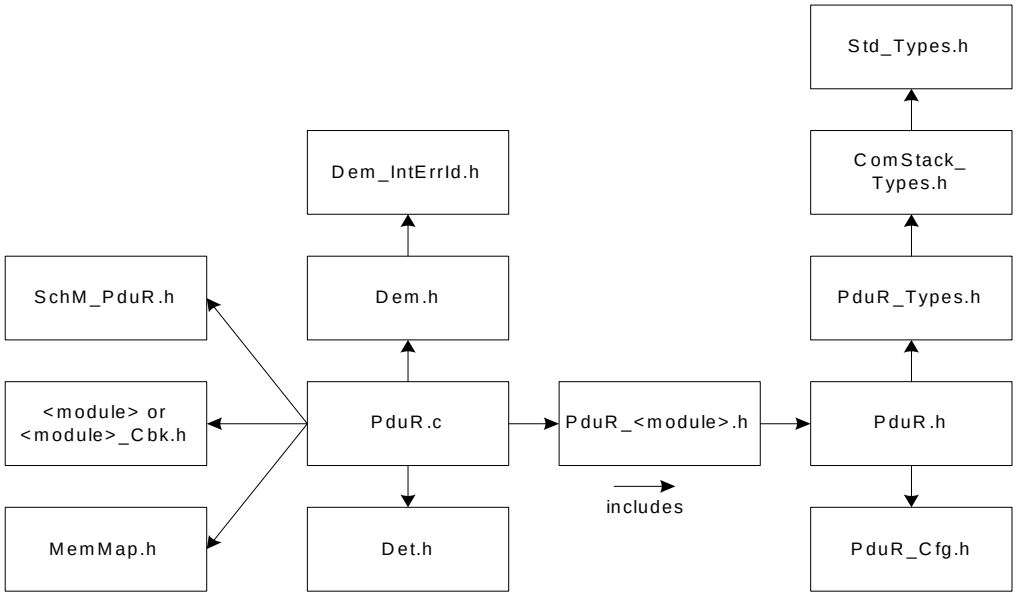

Figure 3: File Structure

This structure allows the separation between platform, compiler and implementation specific definitions and declarations from general definitions as well as the separation of source code and configuration.

By the inclusion of `Dem.h` file the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: General Requirements on Basic Software Modules [3]
Functional Requirements:

Requirement	***Satisfied by***
[BSW00323] API parameter checking	PDUR227, PDUR221, PDUR223, PDUR224
[BSW00336] Shutdown interface	not applicable
[BSW00337] Classification of errors	PDUR100, PDUR101, PDUR331, PDUR332, PDUR103
[BSW00338] Detection and Reporting of development errors	PDUR100, PDUR101, PDUR331, PDUR332, PDUR104, PDUR106, PDUR221, PDUR222, PDUR223, PDUR224, PDUR231
[BSW00339] Reporting of production relevant error status	PDUR103, PDUR232, PDUR233, PDUR100, PDUR255, PDUR258
[BSW00344] Reference to link-time configuration	PDUR240, PDUR226
[BSW00345] Pre-compile-time configuration	PDUR159, PDUR226
[BSW00369] Do not return development error codes via API	PDUR331, PDUR332, chapter 8
[BSW00375] Notification of wake-up reason	not applicable
[BSW00380] Separate C-File for configuration parameters	PDUR226
[BSW00381] Separate configuration header file for pre-compile time parameters	PDUR159
[BSW00383] List dependencies of configuration files	PDUR132
[BSW00384] List dependencies to other modules	chapter 8.5
[BSW00385] List possible error notifications	PDUR100
[BSW00386] Configuration for detecting an error	not applicable
[BSW00387] Specify the configuration class of callback function	chapter 8.5
[BSW00388] Introduce containers	chapter 10
[BSW00389] Containers shall have names	chapter 10.2
[BSW00390] Parameter content shall be unique within the module	chapter 10.2
[BSW00391] Parameter shall have unique names	chapter 10.2
[BSW00392] Parameters shall have a type	chapter 10.2
[BSW00393] Parameters shall have a range	chapter 10.2
[BSW00394] Specify the scope of the parameters	chapter 10.2
[BSW00395] List the required parameters (per parameter)	chapter 10.2
[BSW00396] Configuration classes	chapter 10.2
[BSW00397] Pre-compile-time parameters	chapter 10.2 PDUR242, PDUR243, PDUR245
[BSW00398] Link-time parameters	chapter 10.2 PDUR242
[BSW00399] Loadable Post-build time parameters	chapter 10.2 PDUR244, PDUR246, PDUR247, PDUR248, PDUR249
[BSW004] Version check	Implementation requirement
[BSW00400] Selectable Post-build time parameters	not applicable
[BSW00402] Published information	PDUR236
[BSW00404] Reference to post build time configuration	PDUR241, PDUR226
[BSW00405] Reference to multiple configuration sets	not applicable

Requirement	Satisfied by
[BSW00406] Check module initialization	PDUR324, PDUR325, PDUR326, PDUR327, PDUR119
[BSW00407] Function to read out published parameters	PDUR234
[BSW00409] Header files for production code error IDs	PDUR132, PDUR232
[BSW00412] Separate H-File for configuration parameters	PDUR159
[BSW00416] Sequence of Initialization	not applicable
[BSW00417] Reporting of Error Events by Non-Basic Software	not applicable
[BSW00419] Separate C-Files for pre-compile time configuration parameters	PDUR226
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	not applicable
[BSW00424] BSW main processing function task allocation	not applicable
[BSW00425] Trigger conditions for schedulable objects	not applicable
[BSW00426] Exclusive areas in BSW modules	PDUR214, implementation requirement
[BSW00427] ISR description for BSW modules	not applicable
[BSW00428] Execution order dependencies of main processing functions	not applicable
[BSW00429] Restricted BSW OS functionality access	implementation requirement
[BSW00431] The BSW Scheduler module implements task bodies	implementation requirement
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	not applicable
[BSW00433] Calling of main processing functions	not applicable
[BSW00434] The Schedule Module shall provide an API for exclusive areas	not applicable
[BSW101] Initialization interface	PDUR335, PDUR336, PDUR337
[BSW159] Tool-based configuration	chapter 10
[BSW167] Static configuration checking	PDUR225
[BSW168] Diagnostic Interface of SW components	not applicable
[BSW170] Data for reconfiguration of AUTOSAR SW-components	not applicable
[BSW171] Configurability of optional functionality	PDUR250, PDUR242, PDUR235, chapter 10.2

Document: General Requirements on Basic Software Modules [3]
Selected Non-Functional Requirements:

Requirement	Satisfied by
[BSW00305] Self-defined data types naming convention	PDUR105
[BSW00312] Shared code shall be reentrant	Chapter 7.4
[BSW00346] Basic set of module files	PDUR132, PDUR226
[BSW00379] Module identification	PDUR217
[BSW00415] User dependent include files	PDUR216, PDUR292, PDUR293, PDUR132
[BSW158] Separation of configuration from implementation	PDUR226, PDUR159, PDUR132
[BSW00435] Module Header File Structure for the Basic Software Scheduler	PDUR132

[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	PDUR132

Document: Requirements on Gateway

Requirement	***Satisfied by***
[BSW06001] Protection of routing table	PDUR295, PDUR295
[BSW06002] Updateable Configuration	PDUR295, PDUR295
[BSW06003] Static Routing Rules	PDUR162, PDUR163, PDUR161
[BSW06004] Routing Chronological Order	PDUR297, PDUR298
[BSW06012] Transparent non-TP PDU routing without rate conversion	PDUR160, PDUR166, PDUR168, PDUR436, PDUR437, PDUR193, PDUR194, PDUR448, PDUR449, PDUR450, PDUR451, PDUR463, PDUR464, PDUR465, PDUR466, PDUR452, PDUR453, PDUR467, PDUR468, PDUR201, PDUR237, PDUR430, PDUR431, PDUR303, PDUR304, PDUR305, PDUR306, PDUR307, PDUR252, PDUR253, PDUR310, PDUR311, PDUR255, PDUR256, PDUR312, PDUR313, PDUR258, PDUR259, PDUR308, PDUR309
[BSW06020] PDU Router scalability	PDUR287, PDUR250
[BSW06026] Transparent TP PDU routing	PDUR166, PDUR428, PDUR429, PDUR168, PDUR436, PDUR437, PDUR314, PDUR315, PDUR316, PDUR317, PDUR318, PDUR438, PDUR439, PDUR440, PDUR441, PDUR442, PDUR443, PDUR444, PDUR445, PDUR446, PDUR447, PDUR454, PDUR455, PDUR456, PDUR457, PDUR458, PDUR459, PDUR460, PDUR461, PDUR462, PDUR469, PDUR470, PDUR471, PDUR472, PDUR473, PDUR474, PDUR475, PDUR476, PDUR477, PDUR409, PDUR410, PDUR411, PDUR301, PDUR302, PDUR299, PDUR300
[BSW06029] Routing of Multicast SF-TP PDUs	PDUR164, PDUR299, PDUR300, PDUR301, PDUR302, PDUR314, PDUR315, PDUR316, PDUR317, PDUR318, PDUR438, PDUR439, PDUR440, PDUR441, PDUR442, PDUR445, PDUR446, PDUR447, PDUR454, PDUR455, PDUR456, PDUR457, PDUR460, PDUR461, PDUR462, PDUR469, PDUR470, PDUR471, PDUR472, PDUR475, PDUR476, PDUR477, PDUR206
[BSW06030] Routing of Multicast non TP PDUs without rate conversion	PDUR164, PDUR430, PDUR431, PDUR303, PDUR304, PDUR305, PDUR306, PDUR307, PDUR218, PDUR238
[BSW06032] PDU transmit buffering in PDU Router	PDUR303, PDUR304, PDUR305, PDUR306, PDUR307, PDUR252, PDUR253, PDUR310, PDUR311, PDUR255, PDUR256, PDUR312, PDUR313, PDUR258, PDUR259, PDUR308, PDUR309, PDUR214, PDUR335, PDUR336, PDUR337
[BSW06048] Minimum Routing Capability	PDUR215, PDUR285, PDUR286, PDUR328, PDUR329, PDUR330
[BSW06049] Consistency of PDU Buffer Content	PDUR214
[BSW06097] Configuration identification	PDUR242, PDUR280, PDUR281
[BSW06103] PDU Router Error Handling at unknown PDU-ID	PDUR100, PDUR331, PDUR332, PDUR221
[BSW06104] PDU Router Error Handling at local	PDUR207, PDUR432, PDUR433, PDUR434,

Requirement	*Satisfied by*
reception or transmission	PDUR435
[BSW06105] PDU Router Error Handling in gateway case	PDUR319, PDUR320
[BSW06106] PDU Router Error Handling at FIFO handling	PDUR103, PDUR255, PDUR258
[BSW06119] confirmation in case of fan-out	PDUR301, PDUR302
[BSW06114] PDU Router API for COM	PDUR201, PDUR218, PDUR216
[BSW06115] PDU Router API for DCM	PDUR409, PDUR410, PDUR411, PDUR206, PDUR216
[BSW06116] PDU Router API for IPDUM	PDUR237, PDUR238, PDUR216
[BSW06117] PDU Router API for bus interfaces	PDUR193, PDUR194, PDUR448, PDUR449, PDUR450, PDUR451, PDUR452, PDUR453, PDUR463, PDUR464, PDUR465, PDUR466, PDUR467, PDUR468, PDUR216 PDUR439, PDUR440, PDUR441, PDUR442, PDUR443, PDUR444, PDUR445, PDUR446, PDUR447, PDUR454, PDUR455, PDUR456, PDUR457, PDUR458, PDUR459, PDUR460, PDUR461, PDUR462, PDUR469, PDUR470, PDUR471, PDUR472, PDUR473, PDUR474, PDUR475, PDUR476, PDUR477

7 Functional Specification

The PDU Router module is a PDU transfer unit placed above interface modules and transport protocol modules (lower layer modules) and below COM and DCM (upper layer modules). Beside the PDU Router module is the I-PDU Multiplexer (IPDUM) which provides support for multiplexed I-PDUs. The IPDUM has to be considered as an upper layer module when it calls the PDU Router module to transmit multiplexed I-PDUs or when it is called by the PDU Router module for the reception or transmit confirmation of multiplexed I-PDUs or to provide data via trigger transmit. In case the IPDUM calls the PDU Router module to forward a transmit confirmation or a receive indication to an upper layer (e.g. COM) or when it is called by the PDU Router module to update an I-PDU belonging to a multiplexed I-PDU it has to be considered as lower layer module.

From the ECU point of view, the PDU Router module can perform three different classes of operations:

- PDU Reception: receive I-PDUs and forward them to upper layer modules,
- PDU Transmission: transmit I-PDUs on request of upper layer modules,
- PDU Gateway: (a) receive I-PDUs from an interface module and transmit the I-PDUs immediately via the same or another interface module; or (b) receive I-PDUs from a transport protocol module and transmit the I-PDUs via the same or another transport protocol module.

7.1 General Behavior

PDUR160: The PDU Router module shall transfer an I-PDU without modification to the destination module(s).

PDUR161: The PDU Router module shall identify a PDU uniquely by a static PDU ID.

PDUR162: The PDU Router module's integrator shall define all routes (routing rules) for the PDU Router module in static configuration tables.

PDUR295: The PDU Router module shall support the update of the routing configuration (i.e. the PDU Router routing tables) at post build-time.

PDUR296: The PDU Router module shall update the routing tables only when they are not in use.

Remark: The process how the update of the routing tables is performed is not restricted. Most likely a reflashing of the memory segment that holds the table will be done by the bootloader - a separate program which may be loaded after a reboot to update the ECU.

PDUR281: The post-build time configuration of the PDU Router module shall be identifiable by the unique configuration identifier: PduRConfigurationId.

Remark: The unique configuration identifier is not used to select one of multiple post-build configuration sets of the PDU Router module, but for unique identification of the current PDU Router module post-build configuration, e.g. for Diagnostics or for checking at runtime that the post-build configurations of related communication modules match. The configuration identifier can be read via PduR_GetConfigurationId.

PDUR163: The PDU Router module shall identify the destination(s) of a PDU by using the PDU ID and the static configuration tables.

PDUR297: Only the PDU Router module's environment shall trigger the PDU Router module operation.

PDUR298: The behaviour of all PDU Router module functions shall be synchronous although the overall behavior of a function might be asynchronous (e.g. a transmission request for CAN: PduR_ComTransmit, Com_TxConfirmation).

PDUR164: The PDU Router module shall provide 1:n routing for single frame communication; i.e. (a) I-PDUs to be sent or received via interface modules and (b) I-PDUs to be sent or received within a single frame via TP modules.

PDUR250: The PDU Router module shall allow disabling of optional functionality at pre-compile-time according to the configuration parameters specified by PDUR242. Disabled functionality shall not consume resources (RAM, ROM, runtime).

7.1.1 PDU Reception

PDUR166: For a PDU Reception operation, the PDU Router module shall transfer a received I-PDU from a lower layer module to upper layer module(s) according to the provided PDU ID and the static configuration table.

The receive operation of the PDU Router module shall always be triggered by an indication of a lower layer module (communication interface module, transport protocol module). The indication function is executed by the lower layer either in the context of a cyclic function after polling a communication driver or in the context of an interrupt. In case of the transport protocol module the PDU Router module is requested to provide a receive buffer after the transport protocol module receives a first frame (FF) or single frame (SF) N-PDU. For that purpose the PDU Router module shall forward this request to the related upper layer module by calling <Up>_ProvideRxBuffer. After reception of the last N-PDU the transport protocol module will indicate the PDU Router module that the complete I-PDU has been received and the PDU Router module shall forward this indication to the related upper layer module by calling <Up>_RxIndication. A receive buffer provided by an upper layer module must not be used by the upper layer module until a further buffer is requested or <Up>_RxIndication is called.

PDUR428: When a transport protocol module requests the PDU Router module to provide a receive buffer, the PDU Router module shall forward the request to the related upper layer module by calling <Up>_ProvideRxBuffer.

PDUR429: When a transport protocol module indicates the PDU Router module that it has received the complete I-PDU, the PDU Router module shall forward the indication to the related upper layer module by calling <Up>_RxIndication.

PDUR207: If the receiving TP module reports an error, the PDU Router module shall not perform any error handling and shall simply forward the error to the upper layer module by calling <Up>_RxIndication.

7.1.2 PDU Transmission

PDUR168: For PDU Transmission the PDU Router module shall transfer I-PDUs from an upper layer module to the lower layer module(s) according to the provided PDU ID and the static configuration table.
.

The transmit operation of the PDU Router module shall be triggered by a PDU transmit request from an upper layer module.

PDUR169: The PDU Router module shall forward a request from an upper layer module to the lower layer module(s) according to the PDU ID.

Depending on the used interface module(s) the I-PDU to be transmitted shall be directly provided within the transmit request(s) (i.e. direct data provision) or will later be retrieved by the interface module(s) via the function PduR_<Lo>IfTriggerTransmit (i.e. trigger transmit data provision).

PDUR430: The PDU Router module shall forward request(s) by the interface module(s) via the function PduR_<Lo>IfTriggerTransmit (i.e. trigger transmit data provision) to the upper layer module by calling <Up>_TriggerTransmit.

PDUR431: The mechanism used for each target PDU to transmit an I-PDU to the PDU Router module ID shall be defined in the static configuration tables.

PDUR299: The PDU Router module shall forward a request for a transmission from a TP module to provide a transmit buffer to the related upper layer module by invoking <Up>_ProvideTxBuffer.

PDUR300: In case of a multicast single frame TP transmission, the PDU Router module shall forward only the first transmit buffer request of the TP module to the upper layer module and the PDU Router module shall provide the returned transmit buffer also to the other TP modules.

The transmit operations of the lower layer modules are always asynchronous. This means that a transmission service request returns immediately after the I-PDU has been passed by the PDU Router module to the lower layer module. The PDU Router module will be notified by lower layer modules via PduR_<Lo>IfTxConfirmation or <Lo>TpTxConfirmation respectively after the I-PDU has been transmitted and shall forward this indication to the upper layer module via <Up>_TxConfirmation. The transmit confirmation is always used for TP transmissions and is configurable for unbuffered I-PDUs. A TP transmit buffer provided by an upper layer module may not be used by the upper layer module until a further buffer is requested or <Up>_TxConfirmation is called.

PDUR301: The PDU Router module shall forward the confirmations PduR_<Lo>IfTxConfirmation or <Lo>TpTxConfirmation of lower layer modules to upper layer modules via <Up>_TxConfirmation.

PDUR302: In case of a multicast single frame TP transmission, the PDU Router module shall only forward the transmit confirmation of the last TP module to the upper layer module as the buffer must not be released before.

Lower layer modules will reject a transmission request by returning an error code.

PDUR432: The PDU Router module itself shall not perform any error handling and shall simply return the error code/ return code to the upper layer modules.

PDUR433: In case of a multicast transmission request, the PDU Router module shall return an error code if at least one of the related transmission requests returns an error.

Appropriate error handling is in the responsibility of the upper layer module.

PDUR434: If a transmitting TP module reports an error, the PDU Router module shall not perform any error handling and shall simply forward the error to the upper layer module via <Up>_TxConfirmation.

PDUR435: In case of a multicast single frame TP transmission, the PDU Router module shall forward only the first reported error to the upper layer module in the context of the transmit confirmation of the last TP module.

7.1.3 PDU Gateway

PDUR436: The PDU Router module shall support routing of I-PDUs between communication interface modules or between TP modules (PDU Gateway) that means the PDU Router module shall forward an I-PDU received from one lower layer module (source network) to lower layer modules (destination networks) identified by the provided PDU ID.

PDUR437: The PDU Router module shall support routing of I-PDUs between communication interface modules without rate conversion.

The lower layer modules (communication interface module, transport protocol module) shall trigger the gateway operation of the PDU Router module by transmitting an appropriate I-PDU. They execute the indication functions of the PDU Router module in the context of a cyclic function after polling a communication driver or in the context of an interrupt.

PDUR303: When the PDU Router module receives an I-PDU from a source interface module which shall be forwarded to at least one destination interface module, the PDU Router module shall forward the received I-PDU by calling <Lo>If_Transmit of the destination interface module(s).

PDUR304: The PDU Router module shall provide a dedicated PDU transmit buffer for each destination I-PDU, which is configured to use TriggerTransmit data provision (described by PDUR430, PDUR431).

PDUR307: The PDU Router module shall provide the configuration of the PDU transmit buffer statically as (a) a single buffer with overwrite behavior or as (b) FIFO of size n with flush-on-overrun behavior with the help of configuration parameters PduRSbTxBufferSupport and PduRFifoTxBufferSupport respectively .

PDUR305: When the PDU transmit buffer is configured as FIFO of size n with flush-on-overrun behavior, the PDU Router module shall flush the FIFO in case of a buffer overrun and the new I-PDU shall be used as first entry of the FIFO.

PDUR306: The PDU Router module shall support a FIFO for I-PDUs which is configured to use Direct data provision (even though this is only required in very special cases).

PDUR309: The PDU Router module shall process an internal transmit request for an I-PDU which has a PDU transmit buffer configured as single buffer in the following way:

a) the PDU Router module shall copy the new I-PDU to the transmit buffer
b) the PDU Router module shall call <Lo>If_Transmit of the related interface module
c) the related function call PduR_<Lo>IfTriggerTransmit shall use the I-PDU stored in the transmit buffer

PDUR252: The PDU Router module's integrator shall configure a transmit confirmation for each I-PDU which has a PDU transmit buffer configured as a FIFO (even if it belongs to a multicast transmission).

PDUR253: The PDU Router module shall support two types of FIFOs of PDU transmit buffers: (1) a TT-FIFO for I-PDUs with TriggerTransmit data provision and (2) a D-FIFO for I-PDUs with Direct data provision.

PDUR310: The PDU Router module shall maintain at least two values for each TT-FIFO: (1) the transmit confirmation pending flag (TxConfP), which indicates if a transmit confirmation is pending for the related I-PDU and (2) the index of the current

FIFO entry (TxIdx), which indicates the FIFO entry which shall be used by the next PduR_<Lo>IfTriggerTransmit call.

PDUR255: The PDU Router module shall process an internal transmit request for an I-PDU which has a PDU transmit buffer configured as TT-FIFO in the following way:

a) If TxConfP is not set, the PDU Router module shall replace the new I-PDU with the FIFO entry specified by TxIdx, call <Lo>If_Transmit of the related interface module and set TxConfP if the return code of the function <Lo>If_Transmit is E_OK.
b) If TxConfP is set and the FIFO is not full, the PDU Router module shall add the new I-PDU to the FIFO.
c) If TxConfP is set and the FIFO is full, the PDU Router module shall flush the FIFO (i.e. all entries shall be removed from the FIFO, TxIdx shall be initialized and TxConfP shall be cleared), report the error `PDUR_E_PDU_INSTANCE_LOST` to DEM module and process the new I-PDU according to rule (a).

PDUR256: The PDU Router module shall process a transmit confirmation for an I-PDU which has a PDU transmit buffer configured as TT-FIFO in the following way:

(a) If TxConfP is not set, the PDU Router module shall ignore the confirmation.
(b) If TxConfP is set and the FIFO contains only one entry, the PDU Router module shall set TxConfP to Zero.
(c) If TxConfP is set and the FIFO contains more than one entry, the PDU Router module shall remove the FIFO entry specified by TxIdx, set TxIdx to the next FIFO entry and call <Lo>If_Transmit of the related interface module. If it returns without success (i.e. any value other than E_OK), the PDU Router module shall process the transmit confirmation according to rule (b) or (c).

PDUR312: The PDU Router module shall maintain for each D-FIFO at least a transmit confirmation pending flag (TxConfP) which indicates if a transmit confirmation is pending for the related I-PDU.

PDUR258: The PDU Router module shall process an internal transmit request for an I-PDU which has a PDU transmit buffer configured as D-FIFO in the following way:

(a) If TxConfP is not set, the PDU Router module shall call <Lo>If_Transmit of the related interface module with the new I-PDU. If <Lo>If_Transmit returns with success, the PDU Router module shall set TxConfP, otherwise it shall report the error `PDUR_E_PDU_INSTANCE_LOST` to DEM module.
(b) If TxConfP is set and the FIFO is not full, the PDU Router module shall add the new I-PDU to the FIFO.
(c) If TxConfP is set and the FIFO is full, the PDU Router module shall flush the FIFO (i.e. all entries shall be removed from the FIFO and TxConfP shall be cleared), report the error `PDUR_E_PDU_INSTANCE_LOST` to DEM module and process the new I-PDU according to rule (a).

PDUR259: The PDU Router module shall process a transmit confirmation for an I-PDU which has a PDU transmit buffer configured as D-FIFO in the following way:

(a) If TxConfP is not set, the PDU Router module shall ignore the confirmation.
(b) If TxConfP is set and the FIFO is empty, the PDU Router module shall clear TxConfP.

(c) If TxConfP is set and the FIFO is not empty, the PDU Router module shall call <Lo>If_Transmit of the related interface module with the next FIFO entry. Thereafter, the PDU Router module shall remove this entry from the FIFO. If <Lo>If_Transmit returns without success (i.e. any value other than E_OK), the PDU Router module shall process the transmit confirmation again according to rule (b) or (c).

PDUR478: A remark for the Integrator: The implementation of the critical section has to ensure that a function which entered the critical section will not be preempted by another function which tries to enter the same critical section, i.e. IRQ locks have to be used in case the transmit confirmation PduR_<Lo>IfTxConfirmation is called via an interrupt.

PDUR214: The PDU Router module shall protect the access to PDU transmit buffers by using exclusive areas.

PDUR314: In case of routing between TP modules the PDU Router module shall start forwarding an I-PDU before the full I-PDU is received (“routing-on-the-fly”).

PDUR315: For the routing-on-the-fly the PDU Router module shall provide a small receive buffer when requested via PduR_<Lo>TpProvideRxBuffer.

The buffer size of the receive buffer for routing-on-the-fly shall be equal to the TP block size in case the FrTp retry feature ([11]) is used; for an efficient usage of the buffer, the buffer size should be a multiple of the N-PDU data length. If the provided buffer is smaller than the size of the full I-PDU, the PDU Router module’ environment will call the function PduR_<Lo>TpProvideRxBuffer more than once.

PDUR316: The PDU Router module shall release the previously provided receive buffer for routing-on-the-fly by each call of PduR_<Lo>TpProvideRxBuffer or PduR_<Lo>TpRxIndication and can use it as a transmit buffer for TP transmission on the destination bus.

Hence the usage of a single, large buffer causes store-and-forward routing and the usage of small buffers causes on-the-fly routing.

PDUR317: The PDU Router module shall start the TP transmission on the destination bus by calling <Lo>Tp_Transmit when the first receive buffer is released by the receiving TP module (either within PduR_<Lo>TpProvideRxBuffer or PduR_<Lo>TpRxIndication).

PDUR318: The PDU Router module shall release a related transmit buffer within PduR_<Lo>TpProvideTxBuffer or PduR_<Lo>TpTxConfirmation respectively.

PDUR438: In case of a multicast single frame TP gateway the PDU Router module shall release the buffer for the single frame within PduR_<Lo>TpTxConfirmation when it is called by the last TP module.

Remark: Routing of I-PDUs between communication interface modules with different period or rate (rate conversion) can be done via the COM module. In this case the PDU has to be passed to COM. Based on trigger events COM will decide when to transmit the PDU to the destination communication interface module via the PDU Router. This decision can be derived from the configuration information of the PDU inside the COM module.

PDUR319: The PDU Router module shall not perform any error handling for an I-PDU instance if an interface module rejects a transmit request which belongs to a gateway operation. If no FIFO is configured as PDU transmit buffer, the error shall simply be ignored, otherwise the next FIFO entry shall be used according to PDUR256 and PDUR259 respectively if available.

PDUR320: Whenever a TP module, which is part of an active TP gateway operation, reports an error, the PDU Router module shall stop to continue the TP transmission or TP reception respectively at the related TP modules and shall release the related TP buffers.

7.2 Zero Cost Operation

PDUR287: If the pre-compile time configuration parameter PduRZeroCostOperation is enabled the communication modules directly above or below the PDU Router shall directly call each other without using PDU Router functions (zero cost operation). Therefore the related PDU Router header file shall contain function-like macros which are either evaluated to the related PDU Router function or to the predefined target function (e.g. FrIf_Transmit, CanIf_Transmit) depending on the configuration parameter PduRZeroCostOperation. If PduRZeroCostOperation is enabled then the configuration parameters PduRSingleIf and PduRSingleTp shall be used to specify the related lower layer module and all post-build configuration parameters shall not be used.

7.3 Minimum Routing

PDUR215: The PDU Router module shall provide a minimum routing capability to be able to route specific PDUs from a predefined lower layer interface or TP module to a predefined upper layer module and vice versa without using the post-build time configurable PDU Router routing tables (e.g. access to DCM to bring the ECU into programming mode even when the PDU Router routing tables are corrupted).

Note: PDU Gateway operation, the IPDUM module and multicasts are not supported by minimum routing.

PDUR285: The minimum routing settings shall be separated from the PDU Router routing tables and shall only be configurable at pre-compile time or link-time.

Note: For minimum routing the following (pre-compile time or link time) configuration parameters are used (see PDUR242):

(a) PduRMinimumRoutingUpModule, and PduRMinimumRoutingLoModule to specify the upper and lower layer modules to be used for minimum routing,
(b) PduRMinimumRoutingLoRxPduId and PduRMinimumRoutingUpRxPduId to specify the RxPduIds for PDU Reception and
(c) PduRMinimumRoutingUpTxPduId and PduRMinimumRoutingLoTxPduId to specify the TxPduIds for PDU Transmission.

PDUR286: The PDU Router module shall always precede minimum routing over routing according to the post-build time configurable PDU Router routing tables.

PDUR486: The minimum routing is optional. If any one of the 6 minimum routing Configuration parameter is not configured or left open, then minimum routing feature is not used and no relevant code shall be generated.

In case of zero cost operation, every routing path is implicitely defined and no routing decisions shall be performed by the PDU Router at runtime.

7.4 Reentrancy of API calls

The reentrancy of API calls is generally specified for each API call. If reentrance is allowed then the same API call must not be started with the same PDU ID value while a former call is still ongoing.

7.5 State Management

PDUR324: The PDU Router module shall consist of three states, `PDUR_UNINIT`, `PDUR_REDUCED` and `PDUR_ONLINE` (as shown in Figure 4).

PDUR325: The PDU Router module shall be in the state `PDUR_UNINIT` after power up the PDU Router module.

PDUR326: The PDU Router module shall change to the state `PDUR_ONLINE` when the PDU Router has successfully been initialized via the function PduR_Init.

PDUR327: The PDU Router module shall change to the state `PDUR_REDUCED` in case the initialization within the function PduR_Init did not succeed.

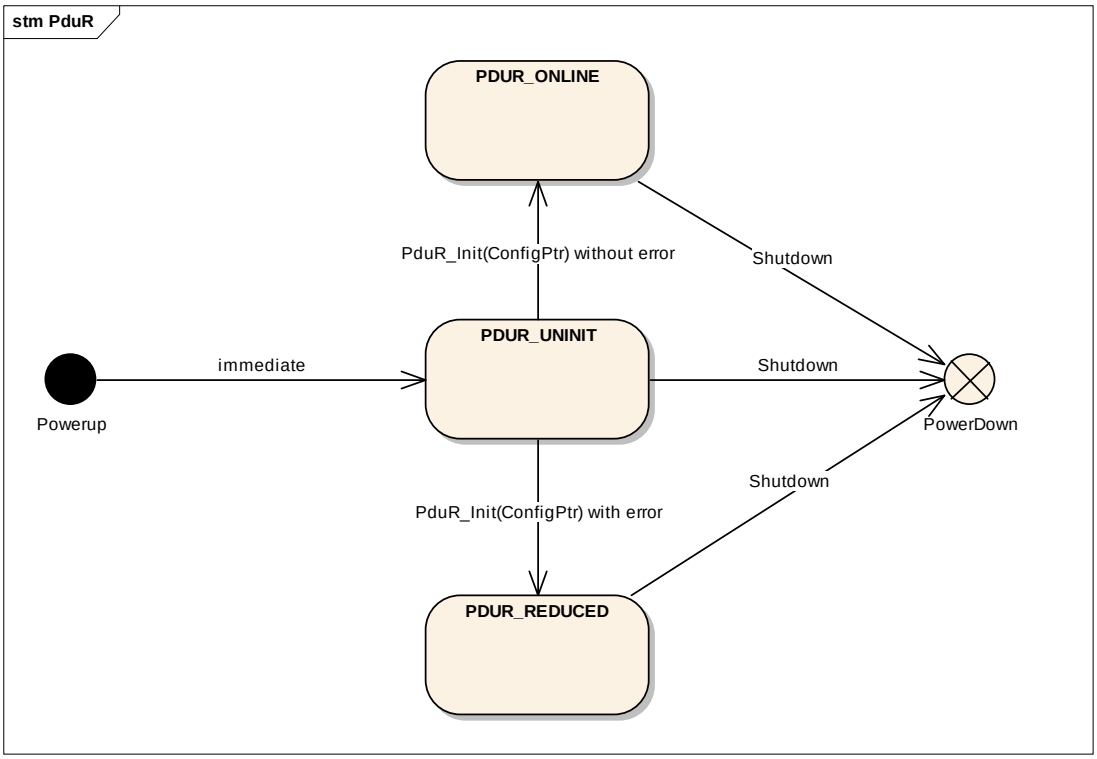

Figure 4: PDU Router states

PDUR328: The PDU Router module shall perform routing of PDUs according to the PDU Router routing tables only when it is in the online state (`PDUR_ONLINE`).

PDUR329: The PDU Router module shall perform minimum routing when it is in the online state (`PDUR_ONLINE`) or reduced state (`PDUR_REDUCED`).

PDUR330: The PDU Router module shall perform no routing when it is in the uninitialised state (`PDUR_UNINIT`).

7.6 Error classification

The general requirements document on AUTOSAR basic software modules [3] distinguish between two types of errors:

(a) errors that can/shall only occur during development and whose detection and/or reporting can be statically configured (on/off)
(b) errors and exceptions that are expected to occur also in production code

PDUR100: The following errors and exceptions shall be detectable by the PDU Router module depending on its build version (development/production mode):

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Invalid configuration pointer	Development	PDUR_E_CONFIG_PTR_INVALID	0x00
API service used without module initialization or PduR_Init called in any state other than PDUR_UNINIT	Development	PDUR_E_INVALID_REQUEST	0x01
Invalid PDU identifier	Development	PDUR_E_PDU_ID_INVALID	0x02
TP module rejects a transmit request for a valid PDU identifier	Development	PDUR_E_TP_TX_REQ_REJECTED	0x03
Data pointer (CanSduPtr, FrSduPtr, LinSduPtr or PduInfoPtr) is NULL	Development	PDUR_E_DATA_PTR_INVALID	0x05
Loss of a PDU instance (FIFO flushed because of an overrun)	Production	PDUR_E_PDU_INSTANCE_LOST	Assigned by DEM
PDU Router initialization failed (PDU Router changed to PDUR_REDUCED state)	Production	PDUR_E_INIT_FAILED	Assigned by DEM

PDUR232: Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.

PDUR231: Development error values are of type uint8.

7.7 Error detection

PDUR101: The detection of development errors is configurable (ON/OFF) at pre-compile time. The switch PduRDevErrorDetect (see chapter 10) shall activate or deactivate the detection of all development errors.

PDUR227: If the PduRDevErrorDetect switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.6 and chapter 7.9.

PDUR233: The detection of production code errors cannot be switched off.
PDUR119: If the PDU Router module has not been initialized (state PDUR_UNINIT), all functions except PduR_Init shall report the error PDUR_E_INVALID_REQUEST via the Development Error Tracer (DET) when called.

7.8 Error notification

PDUR331: Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch PduRDevErrorDetect is set (see chapter 10).

PDUR332: When development error detection is enabled and the PDU Router module has detected an error, it shall report the error to DET module, exit the concerned function and return an error if possible (e.g. by returning `PDUR_E_NOT_OK` in case `Std_ReturnType` is used).

When detecting a development error, the PDU Router module shall report the error to DET by using the DET function shown below:

```
void Det_ReportError(ModuleId, InstanceId, ApiId, ErrorId)
```

`ModuleId`	Module ID of the PDU Router: 51 decimal (see PDUR217)
`InstanceId`	0 (single instance module)
`ApiId`	ID of API which reports an error: Service ID defined in section 8.3
`ErrorId`	ID of detected development error: value according to section 7.6

PDUR103: Production mode errors (see PDUR100) shall be reported to the Diagnostic Event Manager (DEM) by using the DEM function `Dem_ReportErrorStatus(EventId, EventStatus)` specified in [14].

PDUR104: Additional errors that are detected because of specific implementation shall be added in the PDU Router module implementation specification. The classification and enumeration shall be compatible to the errors listed above [PDUR100].

7.9 API parameter checking

PDUR221: If development error detection is enabled, a PDU identifier is not within the specified range, and the PDU identifier is configured to be used by the PDU Router module either for minimum routing (`PDUR_ONLINE` and `PDUR_REDUCED` state) or for routing according to the post-build routing tables (`PDUR_ONLINE` state), the PDU Router module shall report the error `PDUR_E_PDU_ID_INVALID` to the DET module.

PDUR223: If development error detection is enabled and a data pointer (`CanSduPtr`, `FrSduPtr`, `LinSduPtr` or `PduInfoPtr`) is NULL, the PDU Router module shall report the error `PDUR_E_DATA_PTR_INVALID` to the DET module.

PDUR224: If development error detection is enabled and the requested TP buffer size of a gateway operation is larger than the maximum length of all configured TP buffer, the PDU Router module shall report the error `PDUR_E_TP_BUFFER_SIZE_LIMIT` to the DET module.

8 API specification

The following paragraphs specify the API of the PDU Router module.

PDUR217: The Module ID of the PDU Router module shall be 51 (decimal).

8.1 Imported types

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

PDUR333:

Header file	***Imported Type***
Dem_Types.h	Dem_EventIdType
BufReq_Types.h	BufReq_ReturnType
LinIf_Types.h	PduInfoType
PrimitiveTypes.h	PduInfoType
Std_Types.h	Std_VersionInfoType
	Std_ReturnType
FrTp_Types.h	NotifResultType
ComStack_Types.h	PduLengthType
	PduIdType

8.2 Type definitions

PDUR105: The following PDU Router types are specified and shall be defined in `PduR_Types.h`:

8.2.1 PduR_StateType

Name:	`PduR_StateType`	
Type:	`Enumeration`	
Range:	`PDUR_UNINIT`	PDU Router not initialised
	`PDUR_ONLINE`	PDU Router initialized successfully; routing according to minimum routing capability and configurable routing tables
	`PDUR_REDUCED`	PDU Router initialization did not succeed; only minimum routing capability is provided
Description:	States of the PDU Router	

PDUR284: The type PduR_StateType defines the PDU Router states.

8.2.2 PduR_LConfigType

Name:	`PduR_LConfigType`
Type:	`Structure`
Range:	Implementation specific.

Description:	Data structure containing link-time configuration data of the PDU Router

PDUR240: The type PduR_LConfigType is an external data structure containing link-time configuration data of the PDU Router module which shall be implemented in PduR_Lcfg.c if link-time configuration parameters are used (see chapter 5.1.1 and 10.2).

The (optional) link-time configuration allows the configuration of PDU Router features / parameters of a PDU Router module that is provided as object code.

8.2.3 PduR_PBConfigType

Name:	`PduR_PBConfigType`
Type:	`Structure`
Range:	Implementation specific.
Description:	Data structure containing post-build-time configuration data of the PDU Router.

PDUR241: The type PduR_PBConfigType is an external data structure containing post-build-time configuration data of the PDU Router module which shall be implemented in PduR_PBcfg.c (see chapter 5.1.1 and 10.2).

The post-build-time configuration allows the configuration of PDU Router features/ parameters without re-compilation and re-loading of the PDU Router module itself.

8.2.4 PduR_CancelReasonType

PDUR479: PduR_CancelReasonType

Type:	`enum`	
Range:	`PDU_CNLDO`	Cancel Transfer because data are outdated
	`PDU_CNLNB`	Cancel Transfer because no further buffer can be provided
	`PDU_CNLOR`	Cancel Transfer because of another reason
Description:	The reason is sent to the other peer (not on receiver side in a 1:n connection) by the means of an appropriate FC frame.	

8.2.5 PduR_ParameterValueType

PDUR480: PduR_ParameterValueType

Type:	`uint8`	
Range:	0x00 – 0x7F; 0xF1 – 0xF9	PDUR_STmin
Description:	Ranges of this parameter	

8.3 Function definitions

8.3.1 General functions provided by the PDU Router

8.3.1.1 PduR_Init

PDUR334:

Service name:	PduR_Init	
Syntax:	`void PduR_Init( const PduR_PBConfigType* ConfigPtr )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to post build configuration
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the PDU Router	

PDUR335: If the configuration parameter PduRZeroCostOperation is enabled, the function PduR_Init shall be realized as an empty function-like macro.

PDUR336: If the configuration parameter PduRZeroCostOperation is disabled and the PDU Router module is in the state `PDUR_UNINIT`, the function PduR_Init shall initialize the PDU Router module (e.g. PDU transmit buffers shall be initialized according to PDUR308, PDUR309, PDUR310, PDUR311or PDUR312, PDUR313 depending on the PDU transmit buffer type).

PDUR337: If the configuration parameter PduRZeroCostOperation is disabled and the PDU Router module is not in the state `PDUR_UNINIT`, the function PduR_Init shall ignore the request and raise the error `PDUR_E_INVALID_REQUEST` if development error detection is enabled.

PDUR106: The function PduR_Init shall set the state of the PDU Router module to `PDUR_REDUCED` and raise the error `PDUR_E_INIT_FAILED` if the initialization of the PDU Router module failed.

PDUR308: The function PduR_Init shall initialize all PDU transmit buffers which are configured as single buffers through configuration parameter PduRSbTxBufferSupport and copy the the related configured default value to the PDU transmit buffers.

PDUR311: The function PduR_Init shall initialize the PDU transmit buffers which are configured as FIFO through parameter PduRFifoTxBufferSupport, Sets their related transmit confirmation pending flag (TxConfP) to Zero, copy the related configured default value to the FIFO and set their FIFO entry (TxIdx) to this configured default value.

PDUR313: The function PduR_Init shall initialize all D-FIFOs and sets their transmit confirmation pending flag (TxConfP) to Zero.

PDUR222: If development error detection is enabled, the function PduR_Init shall raise development error `PDUR_E_CONFIG_PTR_INVALID` if `ConfigPtr` is NULL.

8.3.1.2 PduR_GetVersionInfo

PDUR338:

Service name:	PduR_GetVersionInfo	
Syntax:	`void PduR_GetVersionInfo( Std_VersionInfoType* versionInfo )`	
Service ID[hex]:	0x17	
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:	Returns the version information of this module.	

PDUR234: The function PduR_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers.

PDUR339: The function PduR_GetVersionInfo shall be realized as a function-like macro when the configuration parameter PduRZeroCostOperation is enabled.

PDUR340: If source code for caller and callee of `PduR_GetVersionInfo` is available, the PDU Router module should realize `PduR _GetVersionInfo` as a macro, defined in the module's header file.

PDUR235: The function PduR_GetVersionInfo shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRVersionInfoApi`

8.3.1.3 PduR_GetConfigurationId

PDUR341:

Service name:	PduR_GetConfigurationId
Syntax:	`uint32 PduR_GetConfigurationId( )`

Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	uint32	Identifier of the post-build time configuration
Description:	Returns the unique identifier of the post-build time configuration of the PDU Router	

PDUR280: The function PduR_GetConfigurationId shall return the unique identifier of the post-build time configuration of the PDU Router module (see PDUR242, PduRConfigurationId) when the configuration parameter PduRZeroCostOperation is disabled.

PDUR342: The function PduR_GetConfigurationId shall be realized as a function-like macro which always returns 0 when the configuration parameter PduRZeroCostOperation is enabled.

8.3.1.4 PduR_CancelTransmitRequest

PDUR481: PduR_CancelTransmitRequest

Service name:	PduR_CancelTransmitRequest	
Syntax:	`Std_ReturnType PduR_CancelTransmitRequest( PduR_CancelReasonType PduCancelReason, PduIdType PduId )`	
Service ID[hex]:	0x1c	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	PduCancelReason	The reason for cancellation
	PduId	This parameter contains the unique identifier of the I-PDU which transfer has to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK Cancellation request of the transfer (sending or receiving) of the specified I-PDU is accepted. E_NOT_OK Cancellation request of the transfer of the specified I-PDU is rejected, e. g. cancellation is requested at the receiver in an 1: n connection or in an unsegmented transfer at the receiver or cancellation is not allowed for the corresponding channel.
Description:	This service primitive is used to cancel the transfer of pending I-PDUs. This function has to be called with the PDU-Id and the reason for cancellation.	

8.3.1.5 PduR_ChangeParameterRequest:

PDUR482: PduR_ChangeParameterRequest

Service name:	PduR_ChangeParameterRequest	
Syntax:	`void PduR_ChangeParameterRequest( PduR_ParameterValueType PduParameterValue, PduIdType PduId )`	
Service ID[hex]:	0x1d	
Sync/Async:	Asynchronous	
Reentrancy:	Non-Reentrant	
Parameters (in):	PduParameterValue	This parameter contains the new value of PDUR_STMIN
	PduId	Gives the ID of the connection (message) for whose channel the change shall be done
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service primitive is used to request the change of the value of the PDUR_STMIN parameter. The new value is given by PduParameterValue. This function has to be called with the PDU-Id and the new value of PDUR_STMIN	

Caveats of PduR_ChangeParameterRequest: According to ISO 15765-2 this is not possible to change the value of the parameter during an ongoing reception.

8.3.2 Function definitions for CAN interaction

8.3.2.1 PduR_CanIfRxIndication

PDUR343:

Service name:	PduR_CanIfRxIndication	
Syntax:	`void PduR_CanIfRxIndication( PduIdType CanRxPduId, const uint8* CanSduPtr )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanRxPduId	ID of CAN L-PDU that has been received. Range: 0..(maximum number of L-PDU IDs which may be received by CAN Interface for the PDU Router) - 1
	CanSduPtr	Pointer to CAN L-SDU (buffer of received payload)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the CAN Interface	

The function PduR_CanIfRxIndication is called by the CAN Interface after a CAN L-

PDU has been received.

PDUR193: The function PduR_CanIfRxIndication shall translate the CanRxPduId into the configured target PDU ID and route this indication to the configured target function. If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a FIFO (PDU transmit buffer), the function PduR_CanIfRxIndication shall process the target PDU according to PDUR258.

PDUR344: The function PduR_CanIfRxIndication shall be callable in interrupt context.

PDUR345: The function PduR_CanIfRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRCanIfSupport.`

8.3.2.2 PduR_CanIfTxConfirmation

PDUR346:

Service name:	PduR_CanIfTxConfirmation	

Syntax:	`void PduR_CanIfTxConfirmation(`
	`  PduIdType CanTxPduId`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanTxPduId	ID of CAN L-PDU that has been transmitted. Range: 0..(maximum number of L-PDU IDs which may be transmitted by CAN Interface) - 1
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the CAN Interface	

The function PduR_CanIfTxConfirmation is called by the CAN Interface after the PDU has been transmitted on the CAN network.

PDUR194: The function PduR_CanIfTxConfirmation shall translate the CanTxPduId into the configured target PDU ID and route this confirmation to the configured target function. If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a FIFO (PDU transmit buffer), the function PduR_CanIfTxConfirmation shall process the confirmation according to PDUR259.

PDUR347: The function PduR_CanIfTxConfirmation shall be callable in interrupt context (e.g. from CAN transmit interrupt). CanIf_Transmit() may be called within PduR_CanIfTxConfirmation.

PDUR348: The function PduR_CanIfTxConfirmation must not be called in the context of CanIf_Transmit.

PDUR349: The function PduR_CanIfTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRCanIfSupport.`

8.3.2.3 PduR_CanTpProvideRxBuffer

PDUR350:

Service name:	PduR_CanTpProvideRxBuffer	
Syntax:	`BufReq_ReturnType PduR_CanTpProvideRxBuffer( PduIdType CanTpRxPduId, PduLengthType TpSduLength, PduInfoType** PduInfoPtr )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanTpRxPduId	ID of CAN N-PDU that shall be received Range: 0..(maximum number of N-PDU IDs which may be received by CAN TP) - 1
	TpSduLength	This length identifies the overall number of bytes to be received. This parameter will not be changed on subsequent calls of this service requesting a new buffer for the same CanTpRxPduId. The length will be greater than zero.
Parameters (inout):	None	
Parameters (out):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a receive buffer. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer available BUFREQ_E_OVFL: Receiver is not able to receive number of TpSduLength bytes; no buffer provided. BUFREQ_E_NOT_OK: Buffer request not successful, no buffer rovided..
Description:	Provides Rx buffer for the CAN TP.	

The function PduR_CanTpProvideRxBuffer is called by the CAN TP for requesting a new buffer (pointer to a PduInfoStructure containing a pointer to a SDU buffer and the buffer length) for the CAN TP to fill in the received data.

PDUR439: The function PduR_CanTpProvideRxBuffer shall translate the CanTpRxPduId into the configured target PDU ID and route this request to the configured target function.

PDUR440: If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the function PduR_CanTpProvideRxBuffer itself has to provide the requested buffer and shall forward the data of the receive buffer to the related receiver(s) when the buffer has been released by a further call of this function.

The length of the buffer, provided by the function PduR_CanTpProvideRxBuffer, does not need to be in the length of the expected SDU. If the returned buffer length is smaller than the expected length the receiver will be requested by a further call of the function PduR_CanTpProvideRxBuffer to provide another buffer, after the current buffer has been filled up with data.

By the function PduR_CanTpProvideRxBuffer the receiver (e.g. DCM) is also informed implicitly about a first frame reception or a single frame reception.

PDUR351: The function PduR_CanTpProvideRxBuffer shall be callable in interrupt context.

PDUR352: The function PduR_CanTpProvideRxBuffer shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRCanTpSupport.`

Caveats of PduR_CanTpProvideRxBuffer: After returning a valid buffer, the receiver must not access this buffer unless:

- it is being requested to provide a new buffer by this service for the same `CanTpRxPduId,` or
- it is being notified by the service PduR_CanTpRxIndication about the successful reception (indication) or
- it is being notified by the service PduR_CanTpRxIndication that the reception was aborted (error indication).

The function PduR_CanTpProvideRxBuffer expects that the CAN TP has transformed the CanRxPduId and the CAN TP related target address information of the TP frame into an ECU-wide unique `CanTpRxPduId`.

8.3.2.4 PduR_CanTpRxIndication

PDUR353:

Service name:	PduR_CanTpRxIndication	

Syntax:	`void PduR_CanTpRxIndication(`
	`  PduIdType CanTpRxPduId,`
	`  NotifResultType Result`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanTpRxPduId	ID of CAN N-PDU that has been received. Range: 0..(maximum number of N-PDU IDs which may be received by CAN TP) - 1
	Result	Result of the TP reception. • NTFRSLT_OK , NTFRSLT_E_CANCELLATION_OK in case TP reception completed successfully • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Cr, NTFRSLT_E_WRONG_SN, NTFRSLT_E_UNEXP_PDU,

		NTFRSLT_E_NO_BUFFER in case TP reception did not complete successfully (e.g. because of a timeout); used to enable unlocking of the receive buffer
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the CAN TP	

The function PduR_CanTpRxIndication is called by the CAN TP.

- with Result = `NTFRSLT_OK` after the complete CAN TP data have successfully been received, i.e. at the very end of the segmented TP receive cycle or after receiving an unsegemented N-PDU.
- with Result != `NTFRSLT_OK` if an error (e.g. timeout) has occurred during the TP reception. This enables unlocking of the receive buffer. It is undefined which part of the buffer contains valid data in this case.

PDUR441: The function PduR_CanTpRxIndication shall translate the CanTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR442: If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the function PduR_CanTpRxIndication shall forward the data of the receive buffer to the related receiver(s), e.g. by using the buffer as a transmit buffer when requested by PduR_<Lo>TpProvideTxBuffer in the gateway case.

PDUR354: The function PduR_CanTpRxIndication shall be callable in interrupt context.

PDUR355: The function PduR_CanTpRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRCanTpSupport`.

8.3.2.5 PduR_CanTpProvideTxBuffer

PDUR356:

Service name:	PduR_CanTpProvideTxBuffer	

Syntax:	`BufReq_ReturnType PduR_CanTpProvideTxBuffer(`
	`  PduIdType CanTpTxPduId,`
	`  PduInfoType** PduInfoPtr,`
	`  uint16 Length`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanTpTxPduId	ID of CAN N-PDU to be transmitted Range: 0..(maximum number of N-PDU IDs which may be transmitted by CAN TP) - 1
	Length	Exact length of the requested transmit buffer; it shall not

		exceed the number of bytes still to be sent. This parameter is needed by the transport protocol to perform error recovery mechanisms. If no error recovery is configured for this PduId, Length may be zero, which indicates that the provided buffer can be of arbitrary size (larger than zero).
Parameters (inout):	None	
Parameters (out):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a transmit buffer. This length must not be smaller than the length given by Length. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer of the requested size is available BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
Description:	Provides Tx buffer for the CAN TP.	

The function PduR_CanTpProvideTxBuffer is called by the CAN TP for requesting a transmit buffer.

The length of the buffer of the function PduR_CanTpProvideTxBuffer does not need to be the length of the complete N-SDU to be transmitted. It only needs to be as large as required by the caller of that service (`Length`).

PDUR443: The function PduR_CanTpProvideTxBuffer shall translate the `CanTpTxPduId` into the configured target PDU ID and route this request to the configured target function.

PDUR444: If `CanTpTxPduId` belongs to a gateway operation the function PduR_CanTpProvideTxBuffer itself has to provide the requested buffer. Therefore the function PduR_CanTpProvideTxBuffer shall use the receive buffer which has previously been filled by the receiving TP module.

PDUR357: The function PduR_CanTpProvideTxBuffer shall be callable in interrupt context.

PDUR358: The function PduR_CanTpProvideTxBuffer shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRCanTpSupport.`

Caveats of PduR_CanTpProvideTxBuffer: In case the function PduR_CanTpProvideTxBuffer returns BUFREQ_E_NOT_OK the related transmit request is not finished. The related TP module may either finish the request by providing a final confirmation indicating an error or may retry the buffer request.

8.3.2.6 PduR_CanTpTxConfirmation

PDUR359:

Service name:	PduR_CanTpTxConfirmation

Syntax:	`void PduR_CanTpTxConfirmation(`
	`  PduIdType CanTpTxPduId,`
	`  NotifResultType Result`
	`)`	

Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanTpTxPduId	ID of CAN N-PDU that has been transmitted. Range: 0..(maximum number of N-PDU IDs which may be transmitted by CAN TP) - 1
	Result	Result of the TP transmission: • NTFRSLT_OK,NTFRSLT_E_CANCELLATION_OK in case TP transmission completed successfully, • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Bs, NTFRSLT_E_INVALID_FS, NTFRSLT_E_WFT_OVRN, NTFRSLT_E_NO_BUFFER in case TP transmission did not complete successfully (e.g. because of a timeout); used to enable unlocking of the transmit buffer.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the CAN TP	

The function PduR_CanTpTxConfirmation is called by the CAN Transport Protocol:

- with Result = `NTFRSLT_OK` after the complete CAN TP data have successfully been transmitted, i.e. at the very end of the segmented TP transmission cycle. This is normally done within the CAN Tx Confirmation interrupt.
- with Result != `NTFRSLT_OK` if an error (e.g. timeout) has occurred during the TP transmission. This enables unlocking of the transmit buffer.

PDUR445: The function PduR_CanTpTxConfirmation shall translate the CanTpTxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR446: If CanTpTxPduId belongs to a gateway operation the function PduR_CanTpTxConfirmation shall use this indication to unlock the transmit buffer.

PDUR447: In case of a multicast single frame TP transmission initiated by an upper layer module the function PduR_CanTpTxConfirmation shall only forward the transmit confirmation of the last TP module to the upper layer module as the buffer must not be released before.

PDUR360: The function PduR_CanTpTxConfirmation shall be callable in interrupt context (e.g. from CAN transmit interrupt).

PDUR361: The function PduR_CanTpTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRCanTpSupport`.

8.3.2.7 PduR_CanTpChangeParameterConfirmation

PDUR483:

Service name:	PduR_CanTpChangeParameterConfirmation	
Syntax:	`void PduR_CanTpChangeParameterConfirmation( PduIdType CanTpTxPduId, NotifResultType Result )`	
Service ID[hex]:	0x1e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	CanTpTxPduId	ID of CAN N-PDU that parameter has been changed. Range: 0..(maximum number of N-PDU IDs which may be transmitted by CAN TP) - 1
	Result	Result of the change parameter: • NTFRSLT_PARAMETER_OK in case parameter changed successfully, • NTFRSLT_E_PARAMETER_NOT_OK, in case Parameter change did not complete successfully.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Change Parameter confirmation for the CAN TP	

The function PduR_CanTpChangeParameterConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PDUR_CANTPCHANGEPARAMETER_SUPPORT`.

8.3.3 Function definitions for FlexRay interaction

8.3.3.1 PduR_FrIfRxIndication

PDUR362:

Service name:	PduR_FrIfRxIndication
Syntax:	`void PduR_FrIfRxIndication( PduIdType FrRxPduId, const uint8* FrSduPtr )`
Service ID[hex]:	0x07
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.

Parameters (in):	FrRxPduId	ID of FlexRay L-PDU that has been received. Range: 0..(maximum number of L-PDU IDs which may be received by FlexRay Interface for the PDU Router) - 1
	FrSduPtr	Pointer to FlexRay SDU (buffer of received payload)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the FlexRay Interface	

The function PduR_FrIfRxIndication is called by the FlexRay Interface after a FlexRay L-PDU has been received.

PDUR448: The function PduR_FrIfRxIndication shall translate the FrRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR449: If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a PDU transmit buffer, the function PduR_FrIfRxIndication shall process the target PDU according to PDUR308, PDUR309, PDUR255 or PDUR258 depending on the PDU transmit buffer type.

PDUR363: The function PduR_FrIfRxIndication shall be callable in interrupt context (e.g. from the FlexRay receive interrupt).

However, the FlexRay specification does not mandate the existence of a receive interrupt.

PDUR364: The function PduR_FrIfRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrIfSupport`.

8.3.3.2 PduR_FrIfTxConfirmation

PDUR365:

Service name:	PduR_FrIfTxConfirmation	

Syntax:	`void PduR_FrIfTxConfirmation(`
	`  PduIdType FrTxPduId`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTxPduId	ID of FlexRay L-PDU that has been transmitted. Range: 0..(maximum number of L-PDU IDs which may be transmitted by FlexRay Interface) - 1
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the FlexRay Interface	

The function PduR_FrIfTxConfirmation is called by the FlexRay Interface after the PDU has been transmitted on the FlexRay network.

PDUR450: The function PduR_FrIfTxConfirmation shall translate the FrTxPduId into the configured target PDU ID and route this confirmation to the configured target function.

PDUR451: If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a FIFO (PDU transmit buffer), the function PduR_FrIfTxConfirmation shall process the confirmation according to PDUR256 or PDUR259 depending on the FIFO buffer type.

PDUR366: The function PduR_FrIfTxConfirmation shall be callable in interrupt context (e.g. from the FlexRay transmit interrupt).

However, since the FlexRay specification does not mandate the existence of a transmit interrupt; the exact meaning of this confirmation (i.e. “transfer into the FlexRay controller’s send buffer” OR “transmission onto the FlexRay network”) depends on the capabilities of the FlexRay communication controller and the configuration of the FlexRay Interface.

PDUR367: The function PduR_FrIfTxConfirmation must not be called in the context of FrIf_Transmit. FrIf_Transmit() may be called within PduR_FrIfTxConfirmation.

PDUR368: The function PduR_FrIfTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrIfSupport`.

8.3.3.3 PduR_FrIfTriggerTransmit

PDUR369:

Service name:	PduR_FrIfTriggerTransmit	
Syntax:	`void PduR_FrIfTriggerTransmit( PduIdType FrTxPduId, uint8* FrSduPtr )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTxPduId	ID of FlexRay L-PDU that is requested to be transmitted. Range: 0..(maximum number of L-PDU IDs which may be transmitted by FlexRay Interface) - 1
	FrSduPtr	Pointer to place inside the transmit buffer of the L-PDU where data shall be copied to.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Triggers the transmission of a FlexRay frame	

The function PduR_FrIfTriggerTransmit is called by the FlexRay Interface for sending out a FlexRay frame. The trigger transmit is initiated by the FlexRay schedule. Whether the function PduR_FrIfTriggerTransmit is called or not is statically configured for each PDU. This triggered transmission is mainly used for the static part of FlexRay.

PDUR452: The function PduR_FrIfTriggerTransmit shall translate the FrTxPduId into the configured target PDU ID and route this trigger to the configured target function (e.g. AUTOSAR COM).

PDUR453: If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a PDU transmit buffer, the function PduR_FrIfTriggerTransmit shall copy the data from the PDU transmit buffer to the place specified by FrSduPtr.

PDUR370: The function PduR_FrIfTriggerTransmit shall be callable in interrupt context.

PDUR371: The function PduR_FrIfTriggerTransmit shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrIfSupport.`

8.3.3.4 PduR_FrTpProvideRxBuffer

PDUR372:

Service name:	PduR_FrTpProvideRxBuffer	

Syntax:	`BufReq_ReturnType PduR_FrTpProvideRxBuffer(`
	`  PduIdType FrTpRxPduId,`
	`  PduLengthType TpSduLength,`
	`  PduInfoType** PduInfoPtr`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTpRxPduId	ID of FlexRay N-PDU that shall be received Range: 0..(maximum number of N-PDU IDs which may be received by FlexRay TP) - 1
	TpSduLength	This length identifies the overall number of bytes to be received. This parameter will not be changed on subsequent calls of this service requesting a new buffer for the same FrTpRxPduId The length will be greater than zero.
Parameters (inout):	None	
Parameters (out):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a receive buffer. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer available

		BUFREQ_E_OVFL: Receiver is not able to receive number of TpSduLength bytes; no buffer provided. BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
Description:	Provides Rx Buffer for the FlexRay TP	

The function PduR_FrTpProvideRxBuffer is called by the FlexRay TP for requesting a new buffer (pointer to a PduInfoStructure containing a pointer to a SDU buffer and the buffer length) for the FlexRay TP to fill in the received data.

PDUR454: The function PduR_FrTpProvideRxBuffer shall translate the FrTpRxPduId into the configured target PDU ID and route this request to the configured target function.

PDUR455: If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case), the function PduR_FrTpProvideRxBuffer itself has to provide the requested buffer and shall forward the data of the receive buffer to the related receiver(s) when the buffer has been released by a further call of this function.

The length of the buffer of the function PduR_FrTpProvideRxBuffer does not need to be in the length of the expected SDU. If the returned buffer length is smaller than the expected length the receiver will be requested by a further call of this service to provide another buffer, after the current buffer has been filled up with data.

By the function PduR_FrTpProvideRxBuffer the receiver (e.g. DCM) is also informed implicitly about a first frame reception or a single frame reception.

PDUR373: The function PduR_FrTpProvideRxBuffer shall be callable in interrupt context.

PDUR374: The function PduR_FrTpProvideRxBuffer shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrTpSupport.`

Caveats of PduR_FrTpProvideRxBuffer: After returning a valid buffer, the receiver must not access this buffer unless:

- it is being requested to provide a new buffer by this service for the same `FrTpRxPduId,` or
- it is being notified by the service PduR_FrTpRxIndication about the successful reception (indication) or
- it is being notified by the service PduR_FrTpRxIndication that the reception was aborted (error indication).

The function PduR_FrTpProvideRxBuffer expects that the FlexRay TP has transformed the FrRxPduId and the FlexRay TP related target address information of the TP frame into an ECU-wide unique `FrTpRxPduId`

8.3.3.5 PduR_FrTpRxIndication

PDUR375:

Service name:	PduR_FrTpRxIndication	
Syntax:	`void PduR_FrTpRxIndication( PduIdType FrTpRxPduId, NotifResultType Result )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTpRxPduId	ID of FlexRay N-PDU that has been received. Range: 0..(maximum number of N-PDU IDs which may be received by FlexRay TP) - 1
	Result	Result of the TP reception. • NTFRSLT_OK , NTFRSLT_E_CANCELLATION_OK in case TP reception completed successfully, • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Cr, NTFRSLT_E_WRONG_SN, NTFRSLT_E_UNEXP_PDU, NTFRSLT_E_NO_BUFFER in case TP reception did not complete successfully (e.g. because of a timeout); used to enable unlocking of the receive buffer
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the FlexRay TP	

The function PduR_FrTpRxIndication is called by the FlexRay

- with Result = `NTFRSLT_OK` after the complete FlexRay TP data have successfully been received, i.e. at the very end of the segmented TP receive cycle or after receiving an unsegemented N-PDU.
- with Result != `NTFRSLT_OK` if an error (e.g. timeout) has occurred during the TP reception. This enables unlocking of the receive buffer. It is undefined which part of the buffer contains valid data in this case.

PDUR456: The function PduR_FrTpRxIndication shall translate the FrTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR457: If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case), the function PduR_FrTpRxIndication shall forward the data of the receive buffer to the related receiver(s), e.g. by using the buffer as a transmit buffer when requested by PduR_<Lo>TpProvideTxBuffer in the gateway case.

PDUR376: The function PduR_FrTpRxIndication shall be callable in interrupt context.

PDUR377: The function PduR_FrTpRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrTpSupport.`

8.3.3.6 PduR_FrTpProvideTxBuffer

PDUR378:

Service name:	PduR_FrTpProvideTxBuffer	
Syntax:	`BufReq_ReturnType PduR_FrTpProvideTxBuffer( PduIdType FrTpTxPduId, PduInfoType** PduInfoPtr, uint16 Length )`	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTpTxPduId	ID of FlexRay N-PDU to be transmitted. Range: 0..(maximum number of N-PDU IDs which may be transmitted by FlexRay TP) - 1
	Length	Exact length of the requested transmit buffer; it shall not exceed the number of bytes still to be sent. This parameter is needed by the transport protocol to perform error recovery mechanisms. If no error recovery is configured for this PduId, Length may be zero, which indicates that the provided buffer can be of arbitrary size (larger than zero).
Parameters (inout):	None	
Parameters (out):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a transmit buffer. This length must not be smaller than the length given by Length. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer of the requested size is available BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
Description:	ProvidesTx Buffer for the FlexRay TP	

The function PduR_FrTpProvideTxBuffer is called by the FlexRay TP for requesting a transmit buffer.

The length of the buffer of the function PduR_FrTpProvideTxBuffer does not need to be the length of the complete N-SDU to be transmitted. It only needs to be as large as required by the caller of that service (`Length`).

PDUR458: The function PduR_FrTpProvideTxBuffer shall translate the `FrTpTxPduId` into the configured target PDU ID and route this request to the configured target function.

PDUR459: If `FrTpTxPduId` belongs to a gateway operation the function PduR_FrTpProvideTxBuffer itself has to provide the requested buffer. Therefor the function PduR_FrTpProvideTxBuffer shall use the reveive buffer which has previously been filled by the receiving TP module.

PDUR379: The function PduR_FrTpProvideTxBuffer shall be callable in interrupt context.

PDUR380: The function PduR_FrTpProvideTxBuffer shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrTpSupport.`

Caveats of PduR_FrTpProvideTxBuffer: In case the function PduR_FrTpProvideTxBuffer returns BUFREQ_E_NOT_OK the related transmit request is not finished. The related TP module may either finish the request by providing a final confirmation indicating an error or may retry the buffer request.

8.3.3.7 PduR_FrTpTxConfirmation

PDUR381:

Service name:	PduR_FrTpTxConfirmation	
Syntax:	`void PduR_FrTpTxConfirmation( PduIdType FrTpTxPduId, NotifResultType Result )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTpTxPduId	ID of FlexRay N-PDU that has been transmitted. Range: 0..(maximum number of N-PDU IDs which may be transmitted by FlexRay TP) - 1
	Result	Result of the TP transmission: • NTFRSLT_OK, NTFRSLT_E_CANCELLATION_OK in case TP transmission completed successfully, • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Bs, NTFRSLT_E_INVALID_FS, NTFRSLT_E_WFT_OVRN, NTFRSLT_E_NO_BUFFER in case TP transmission did not

		complete successfully (e.g. because of a timeout); used to enable unlocking of the transmit buffer.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the FlexRay TP	

The function PduR_FrTpTxConfirmation is called by the FlexRay TP:

- with Result = `NTFRSLT_OK` after the complete FlexRay TP data have successfully been transmitted, i.e. at the very end of the segmented TP transmission cycle.
- with Result != `NTFRSLT_OK` if an error (e.g. timeout) has occurred during the TP transmission. This enables unlocking of the transmit buffer.

PDUR460: The function PduR_FrTpTxConfirmation shall translate the FrTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR461: If FrTpRxPduId belongs to a gateway operation the function PduR_FrTpTxConfirmation shall use this indication to unlock the transmit buffer.

PDUR462: In case of a multicast single frame TP transmission initiated by an upper layer module, the function PduR_FrTpTxConfirmation shall only forward the transmit confirmation of the last TP module to the upper layer module as the buffer must not be released before.

PDUR382: The function PduR_FrTpTxConfirmation shall be callable in interrupt context (e.g. from FlexRay transmit interrupt).

However, since the FlexRay Specification does not mandate the existence of a transmit interrupt, the exact meaning of the confirmation within the function PduR_FrTpTxConfirmation (i.e. “transfer into the FlexRay controller’s send buffer” OR “transmission onto the FlexRay network”) depends on the capabilities of the FlexRay communication controller and the configuration of the FlexRay Interface.

PDUR383: The function PduR_FrTpTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRFrTpSupport.`

8.3.3.8 PduR_FrTpChangeParameterConfirmation

PDUR484:

Service name:	PduR_FrTpChangeParameterConfirmation	
Syntax:	`void PduR_FrTpChangeParameterConfirmation( PduIdType FrTpTxPduId, NotifResultType Result )`	
Service ID[hex]:	0x1f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	FrTpTxPduId	ID of FlexRay N-PDU that parameter has been changed.

		Range: 0..(maximum number of N-PDU IDs which may be transmitted by FlexRay TP) - 1
	Result	Result of the change parameter: • NTFRSLT_PARAMETER_OK in case parameter changed successfully, • NTFRSLT_E_PARAMETER_NOT_OK, in case Parameter change did not complete successfully.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Change Parameter confirmation for the FlexRay TP	

The function PduR_FrTpChangeParameterConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PDUR_FRTPCHANGEPARAMETER_SUPPORT`.

8.3.4 Function definitions for LIN interaction

8.3.4.1 PduR_LinIfRxIndication

PDUR384:

Service name:	PduR_LinIfRxIndication	

Syntax:	`void PduR_LinIfRxIndication(`
	`  PduIdType LinRxPduId,`
	`  const uint8* LinSduPtr`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinRxPduId	ID of LIN L-PDU that has been received. Range: 0..(maximum number of L-PDU IDs which may be received by LIN Interface for the PDU Router) - 1
	LinSduPtr	Pointer to LIN L-SDU (buffer of received payload)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the LIN Interface	

The function PduR_LinIfRxIndication is called by the LIN Interface after a LIN L-PDU has been received.

PDUR463: The function PduR_LinIfRxIndication shall translate the LinRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR464: If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a PDU transmit buffer, the function PduR_LinIfRxIndication shall process the target PDU according to PDUR308, PDUR309, PDUR255 or PDUR258 depending on the PDU transmit buffer type.

PDUR385: The function PduR_LinIfRxIndication shall be callable in interrupt context.

PDUR386: The function PduR_LinIfRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinIfSupport`.

8.3.4.2 PduR_LinIfTxConfirmation

PDUR387:

Service name:	PduR_LinIfTxConfirmation	
Syntax:	`void PduR_LinIfTxConfirmation( PduIdType LinTxPduId )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTxPduId	ID of LIN L-PDU that has been transmitted. Range: 0..(maximum number of L-PDU IDs which may be transmitted by LIN Interface) - 1
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the LIN Interface	

The function PduR_LinIfTxConfirmation is called by the LIN Interface after the PDU has been transmitted on the LIN bus.

PDUR465: The function PduR_LinIfTxConfirmation shall translate the LinTxPduId into the configured target PDU ID and route this confirmation to the configured target function.

PDUR466: If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a FIFO (PDU transmit buffer), the function PduR_LinIfTxConfirmation shall process the confirmation according to PDUR256 or PDUR259 depending on the FIFO buffer type.

PDUR388: The function PduR_LinIfTxConfirmation shall be callable in interrupt context.

PDUR389: The function PduR_LinIfTxConfirmation must not be called in the context of LinIf_Transmit. LinIf_Transmit() may be called within PduR_LinIfTxConfirmation.

PDUR390: The function PduR_LinIfTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinIfSupport.`

8.3.4.3 PduR_LinIfTriggerTransmit

PDUR391:

Service name:	PduR_LinIfTriggerTransmit	
Syntax:	`void PduR_LinIfTriggerTransmit( PduIdType LinTxPduId, uint8* LinSduPtr )`	
Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTxPduId	ID of LIN L-PDU that is requested to be transmitted. Range: 0..(maximum number of L-PDU IDs which may be transmitted by LIN Interface) - 1
	LinSduPtr	Pointer to place inside the transmit buffer of the L-PDU where data shall be copied to.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Triggers the transmission of a LIN frame	

The function PduR_LinIfTriggerTransmit is called by the LIN Master for sending out a LIN frame. The trigger transmit can be initiated by the Master schedule table itself or a received LIN header. Whether this function is called or not is statically configured for each PDU.

PDUR467: The function PduR_LinIfTriggerTransmit shall translate the LinTxPduId into the configured target PDU ID and route this trigger to the configured target function (e.g. AUTOSAR COM).

PDUR468: If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a PDU transmit buffer, the function PduR_LinIfTriggerTransmit shall copy the data from the PDU transmit buffer to the place specified by LinSduPtr.

PDUR392: The function PduR_LinIfTriggerTransmit shall be callable in interrupt context.

PDUR393: The function PduR_LinIfTriggerTransmit shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinIfSupport.`

8.3.4.4 PduR_LinTpProvideRxBuffer

PDUR394:

Service name:	PduR_LinTpProvideRxBuffer	
Syntax:	`BufReq_ReturnType PduR_LinTpProvideRxBuffer( PduIdType LinTpRxPduId, PduLengthType TpSduLength, PduInfoType** PduInfoPtr )`	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTpRxPduId	ID of LIN N-PDU that shall be received Range: 0..(maximum number of N-PDU IDs which may be received by LIN TP) - 1
	TpSduLength	This length identifies the overall number of bytes to be received. This parameter will not be changed on subsequent calls of this service requesting a new buffer for the same LinTpRxPduId. The length will be greater than zero.
Parameters (inout):	None	
Parameters (out):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a receive buffer. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer available BUFREQ_E_OVFL: Receiver is not able to receive number of TpSduLength bytes; no buffer provided. BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
Description:	Provides Rx Buffer for the LIN TP	

The function PduR_LinTpProvideRxBuffer is called by the LIN TP for requesting a new buffer (pointer to a PduInfoStructure containing a pointer to a SDU buffer and the buffer length) for the LIN TP to fill in the received data.

PDUR469: The function PduR_LinTpProvideRxBuffer shall translate the LinTpRxPduId into the configured target PDU ID and route this request to the configured target function.

PDUR470: If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case), the function PduR_LinTpProvideRxBuffer itself has to provide the requested buffer and shall forward the data of the receive buffer to the related receiver(s) when the buffer has been released by a further call of this function.

The length of the buffer of the function PduR_LinTpProvideRxBuffer does not need to be in the length of the expected SDU. If the returned buffer length is smaller than the expected length the receiver will be requested by a further call of this service to provide another buffer, after the current buffer has been filled up with data.

By the function PduR_LinTpProvideRxBuffer the receiver (e.g. DCM) is also informed implicitly about a first frame reception or a single frame reception.

PDUR395: The function PduR_LinTpProvideRxBuffer shall be callable in interrupt context.

PDUR396: The function PduR_LinTpProvideRxBuffer shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinTpSupport.`

Caveats of PduR_LinTpProvideRxBuffer: After returning a valid buffer, the receiver must not access this buffer unless:

- it is being requested to provide a new buffer by this service for the same `LinTpRxPduId,` or
- it is being notified by the service PduR_LinTpRxIndication about the successful reception (indication), or
- it is being notified by the service PduR_LinTpRxIndication that the reception was aborted (error indication).

The function PduR_LinTpProvideRxBuffer expects that the LIN TP has transformed the LinRxPduId and the LIN TP related target address information of the TP frame into an ECU-wide unique LinTpRxPduId.

8.3.4.5 PduR_LinTpRxIndication

PDUR397:

Service name:	PduR_LinTpRxIndication	
Syntax:	`void PduR_LinTpRxIndication( PduIdType LinTpRxPduId, NotifResultType Result )`	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTpRxPduId	ID of LIN N-PDU that has been received. Range: 0..(maximum number of N-PDU IDs which may be received by LIN TP) - 1
	Result	Result of the TP reception. • NTFRSLT_OK , NTFRSLT_E_CANCELLATION_OK in case TP reception completed successfully; • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Cr, NTFRSLT_E_WRONG_SN, NTFRSLT_E_UNEXP_PDU,

		NTFRSLT_E_NO_BUFFER in case TP reception did not complete successfully (e.g. because of a timeout); used to enable unlocking of the receive buffer
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the LIN TP	

The function PduR_LinTpRxIndication is called by the LIN TP

- with Result = `NTFRSLT_OK` after the complete LIN TP data have successfully been received, i.e. at the very end of the segmented TP receive cycle or after receiving an unsegemented N-PDU.
- with Result != `NTFRSLT_OK` if an error (e.g. timeout) has occurred during the TP reception. This enables unlocking of the receive buffer. It is undefined which part of the buffer contains valid data in this case.

PDUR471: The function PduR_LinTpRxIndication shall translate the LinTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR472: If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the function PduR_LinTpRxIndication shall forward the data of the receive buffer to the related receiver(s), e.g. by using the buffer as a transmit buffer when requested by PduR_<Lo>TpProvideTxBuffer in the gateway case.

PDUR398: The function PduR_LinTpRxIndication shall be callable in interrupt context.

PDUR399: The function PduR_LinTpRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinTpSupport.`

8.3.4.6 PduR_LinTpProvideTxBuffer

PDUR400:

Service name:	PduR_LinTpProvideTxBuffer	

Syntax:	`BufReq_ReturnType PduR_LinTpProvideTxBuffer(`
	`  PduIdType LinTpTxPduId,`
	`  PduInfoType** PduInfoPtr,`
	`  uint16 Length`
	`)`	

Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTpTxPduId	ID of LIN N-PDU to be transmitted Range: 0..(maximum number of N-PDU IDs which may be transmitted by LIN TP) - 1
	Length	Exact length of the requested transmit buffer; it shall not

		exceed the number of bytes still to be sent. This parameter is needed by the transport protocol to perform error recovery mechanisms. If no error recovery is configured for this PduId, Length may be zero, which indicates that the provided buffer can be of arbitrary size (larger than zero).
Parameters (inout):	None	
Parameters (out):	PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a transmit buffer. This length must not be smaller than the length given by Length. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer of the requested size is available BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
Description:	Provides Tx Buffer for the LIN TP	

The function PduR_LinTpProvideTxBuffer is called by the LIN TP for requesting a transmit buffer.

The length of the buffer of the function PduR_LinTpProvideTxBuffer does not need to be in the length of the complete N-SDU to be transmitted. It only needs to be as large as required by the caller of that service (`Length`).

PDUR473: The function PduR_LinTpProvideTxBuffer shall translate the `LinTpTxPduId` into the configured target PDU ID and route this request to the configured target function.

PDUR474: If `LinTpTxPduId` belongs to a gateway operation the function PduR_LinTpProvideTxBuffer itself has to provide the requested buffer. Therefor the function PduR_LinTpProvideTxBuffer shall use the receive buffer which has previously been filled by the receiving TP module.

PDUR401: The function PduR_LinTpProvideTxBuffer shall be callable in interrupt context.

In case function PduR_LinTpProvideTxBuffer returns BUFREQ_E_NOT_OK the related transmit request is not finished. The related TP module may either finish the request by providing a final confirmation indicating an error or may retry the buffer request.

PDUR402: The function PduR_LinTpProvideTxBuffer shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinTpSupport.`

8.3.4.7 PduR_LinTpTxConfirmation

PDUR403:

Service name:	PduR_LinTpTxConfirmation	
Syntax:	`void PduR_LinTpTxConfirmation( PduIdType LinTpTxPduId, NotifResultType Result )`	
Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTpTxPduId	ID of LIN N-PDU that has been transmitted. Range: 0..(maximum number of N-PDU IDs which may be transmitted by LIN TP) - 1
	Result	Result of the TP transmission: • NTFRSLT_OK, NTFRSLT_E_CANCELLATION_OK in case TP transmission completed successfully, • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Bs, NTFRSLT_E_INVALID_FS, NTFRSLT_E_WFT_OVRN, NTFRSLT_E_NO_BUFFER in case TP transmission did not complete successfully (e.g. because of a timeout); used to enable unlocking of the transmit buffer.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the LIN TP	

The function PduR_LinTpTxConfirmation is called by the LIN TP:

- with Result = `NTFRSLT_OK` after the complete LIN TP data have successfully been transmitted, i.e. at the very end of the segmented TP transmission cycle.
- with Result != `NTFRSLT_OK` if an error (e.g. timeout) has occurred during the TP transmission. This enables unlocking of the transmit buffer.

PDUR475: The function PduR_LinTpTxConfirmation shall translate the LinTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

PDUR476: If LinTpRxPduId belongs to a gateway operation the function PduR_LinTpTxConfirmation shall use this indication to unlock the transmit buffer.

PDUR477: In case of a multicast single frame TP transmission initiated by an upper layer module the function PduR_LinTpTxConfirmation shall only forward the transmit confirmation of the last TP module to the upper layer module as the buffer must not be released before.

PDUR404: The function PduR_LinTpTxConfirmation shall be callable in interrupt context.

PDUR405: The function PduR_LinTpTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRLinTpSupport.`

8.3.4.8 PduR_LinTpChangeParameterConfirmation

PDUR485:

Service name:	PduR_LinTpChangeParameterConfirmation	
Syntax:	`void PduR_LinTpChangeParameterConfirmation( PduIdType LinTpTxPduId, NotifResultType Result )`	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	LinTpTxPduId	ID of LIN N-PDU that parameter has been changed. Range: 0..(maximum number of N-PDU IDs which may be transmitted by LIN TP) - 1
	Result	Result of the change parameter: • NTFRSLT_PARAMETER_OK in case parameter changed successfully, • NTFRSLT_E_PARAMETER_NOT_OK, in case Parameter change did not complete successfully.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Change Parameter confirmation for the LIN TP	

The function PduR_CanTpChangeParameterConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PDUR_LINTPCHANGEPARAMETER_SUPPORT.`

8.3.5 Function definitions for COM interaction

8.3.5.1 PduR_ComTransmit

PDUR406:

Service name:	PduR_ComTransmit	
Syntax:	`Std_ReturnType PduR_ComTransmit( PduIdType ComTxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	ComTxPduId	ID of AUTOSAR COM I-PDU to be transmitted. Range: 0..(maximum number of I-PDU IDs which may be transmitted by COM) - 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 has been accepted E_NOT_OK: Transmit request has not been accepted
Description:	Requests a tansmission for the AUTOSAR COM Module	

The function PduR_ComTransmit is called by AUTOSAR COM to request a transmission.

PDUR201: The function PduR_ComTransmit shall translate the ComTxPduId into the configured target PDU ID and route this transmit request to the configured target interface module.

PDUR218: If ComTxPduId represents a group of PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error, the function PduR_ComTransmit shall return E_NOT_OK.

PDUR407: The function PduR_ComTransmit shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRComSupport`.

8.3.6 Function definitions for DCM interaction

8.3.6.1 PduR_DcmTransmit

PDUR408:

Service name:	PduR_DcmTransmit	
Syntax:	`Std_ReturnType PduR_DcmTransmit( PduIdType DcmTxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	DcmTxPduId	ID of DCM I-PDU to be transmitted. Range: 0..(maximum number of I-PDU IDs which may be transmitted by DCM) - 1
	PduInfoPtr	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 has been accepted E_NOT_OK: Transmit request has not been accepted
Description:	Requests a transmission for the DCM module.	

The function PduR_DcmTransmit is called by the DCM to request a transmission.

PDUR409: The function PduR_DcmTransmit shall translate the DcmTxPduId into the configured target PDU ID and route this transmit request to the configured target TP module.

PDUR410: The function PduR_DcmTransmit shall only use the SduLength information within the parameter PduInfoPtr.

PDUR411: The function PduR_DcmTransmit must not use the parameter PduInfoPtr because it is a pointer to undefined data

For a TP transmission request the function call PduR_DcmTransmit will be followed by at least one invocation of PduR_<Lo>TpProvideTxBuffer by the TP module to get the data (transmission buffer).
The reason for having the PduInfoPtr is to reach compliance with the COM API PduR_ComTransmit().

PDUR206: If the parameter DcmTxPduId of the function PduR_DcmTransmit represents a group of single frame TP PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error, the function PduR_DcmTransmit shall return E_NOT_OK.

PDUR412: The function PduR_DcmTransmit shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRDcmSupport`.

8.3.7 Function definitions for IPDUM interaction

8.3.7.1 PduR_IpdumTransmit

PDUR413:

Service name:	PduR_IpdumTransmit	
Syntax:	`Std_ReturnType PduR_IpdumTransmit( PduIdType IpdumTxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	IpdumTxPduId	ID of IPDUM I-PDU to be transmitted. Range: 0..(maximum number of I-PDU IDs which may be transmitted by IPDUM) – 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 has been accepted E_NOT_OK: Transmit request has not been accepted
Description:	Requests a transmission for the IPDUM	

The function PduR_IpdumTransmit is called by IPDUM (acting as an upper layer module) to request a transmission on a lower layer module (e.g. CanIf, FrIf, LinIf).

PDUR237: The function PduR_IpdumTransmit shall translate the IpdumUpTxPduId into the configured target PDU ID and route this transmit request to the configured target interface module.

PDUR238: If the parameter IpdumUpTxPduId of the function PduR_IpdumTransmit represents a group of PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error, the function PduR_IpdumTransmit shall return E_NOT_OK.

PDUR414: The function PduR_IpdumTransmit shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRIPduMSupport.`

8.3.7.2 PduR_IpdumTxConfirmation

PDUR415:

Service name:	PduR_IpdumTxConfirmation
Syntax:	`void PduR_IpdumTxConfirmation( PduIdType IpdumLoTxPduId )`

Service ID[hex]:	0x1a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	IpdumLoTxPduId	ID of IPDUM I-PDU to be transmitted. Range: 0..(maximum number of I-PDU IDs which may be transmitted by IPDUM) – 1
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Tx confirmation for the IPDUM	

The function PduR_IpdumTxConfirmation is called by IPDUM (acting as a lower layer module) after the PDU has been transmitted.

PDUR416: The function PduR_IpdumTxConfirmation shall translate the IpdumLoTxPduId into the configured target PDU ID and route this confirmation to the configured upper layer module (e.g. COM).

PDUR417: The function PduR_IpdumTxConfirmation shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRIPduMSupport.`

PDUR418: The function PduR_IpdumTxConfirmation shall be callable in interrupt context.

8.3.7.3 PduR_IpdumRxIndication

PDUR419:

Service name:	PduR_IpdumRxIndication	

Syntax:	`void PduR_IpdumRxIndication(`
	`  PduIdType IpdumLoRxPduId,`
	`  const uint8* IpdumSduPtr`
	`)`	

Service ID[hex]:	0x1b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	IpdumLoRxPduId	ID of IPDUM I-PDU that has been received. Range: 0..(maximum number of I-PDU IDs which may be received by IPDUM) – 1
	IpdumSduPtr	Pointer to IPDUM SDU (buffer of received payload)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Rx indicator for the IPDUM	

The function PduR_IpdumRxIndication is called by the IPDUM (acting as a lower layer module) after the PDU has been received.

PDUR420: The function PduR_IpdumRxIndication shall translate the parameter IpdumLoRxPduId into the configured target PDU ID and route this indication to the configured upper layer module (e.g. COM).

PDUR421: The function PduR_IpdumRxIndication shall be callable in interrupt context.

PDUR422: The function PduR_IpdumRxIndication shall be pre-compile time configurable `On/Off` by the configuration parameter: `PduRIPduMSupport.`

8.4 Scheduled functions

As any PDU Router operation is triggered by an adjacent communication module the PDU Router does not require scheduled functions.

8.5 Expected Interfaces

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

8.5.1 Mandatory Interfaces

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

PDUR423:

API function	*Description*
Dem_ReportErrorStatus	Reports errors to the DEM.

8.5.2 Optional Interfaces

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

PDUR424:

API function	*Description*
Ipdum_TxConfirmation	This function is called by the lower layer after the I-PDU has been transmitted on the network.
CanTp_Transmit	This service is used to request the transfer of segmented data.
Det_ReportError	Service to report development errors.
LinTp_Transmit	Requests the transfer of segmented data.
FrIf_Transmit	Requests the sending of a PDU.

FrTp_Transmit	This service is utilized to request the transfer of data. It sets a flag for indicating that a transmit request is present. This function has to be called with the PDU-Id of the FrTp, i.e. the upper layer has to translate its own PDU-Id into the one of the TP for the corresponding message. Within the provided PduInfoPtr only SduLength is valid (no data)! If this function returns E_OK then there will arise an call of PduR_FrTpProvideTxBuffer in order to get data for sending.
Ipdum_Transmit	Service is called by the PDU-Router to request a transmission.
Com_TxConfirmation	This function is called by the lower layer after the PDU has been transmitted on the network. A confirmation that is received for an I-PDU that does not require a confirmation is silently discarded.
Com_TriggerTransmit	This function is called by the lower layer when an AUTOSAR COM I-PDU shall be transmitted. Within this function, AUTOSAR COM shall copy the contents of its I-PDU transmit buffer to the L-PDU buffer given by SduPtr.
Com_RxIndication	This function is called by the lower layer after an I-PDU has been received.
Dcm_RxIndication	Function Definitions
Ipdum_RxIndication	Service is called when a multiplexed SDU has to be de-multiplexed.
Ipdum_TriggerTransmit	Service is called by the lower layer when an IPduM I-PDU shall be transmitted.
CanIf_Transmit	--
Dcm_ProvideTxBuffer	--
Dcm_TxConfirmation	--
Dcm_ProvideRxBuffer	--

8.5.3 Configurable interfaces

The PDU Router does not provide interfaces where the target function could be configured.

9 Sequence diagrams

The goal of this chapter is to make the understanding of the PDU Router easier. For this purpose sequence diagrams which show different communication scenarios are used. Please consider that the sequence diagrams are not exhaustive and are only used to support the functional specification (chapter 7) and API specification (chapter 8).

Focus of the sequence diagrams is the PDU Router and therefore interactions between other modules (e.g. between an interface and its driver) are not shown. The sequence diagrams are grouped in four subchapters: Initialization (9.1), PDU Reception (9.2), PDU Transmission (9.3) and PDU Gateway (9.4).

Note: The sequence diagrams of the I-PDU Mulitiplexer are shown in [17]. Depending on the interaction scenario the IPDUM has to be considered as an upper layer or a lower layer module of the PDU Router.

9.1 Initialization

The initialization of the PDU Router is shown by Figure 5.

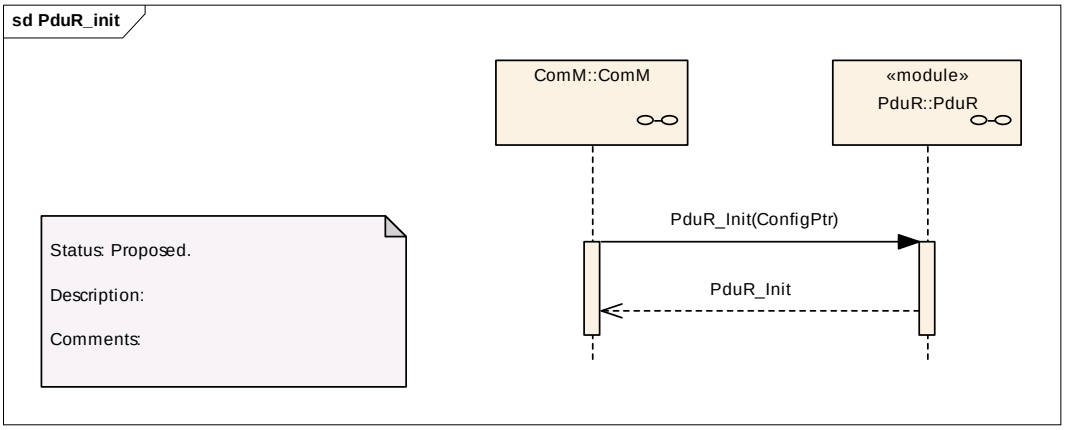

Figure 5: Initialization

9.2 PDU Reception

The reception of an I-PDU received from an interface module (non-TP PDU Rx) is shown by Figure 6.

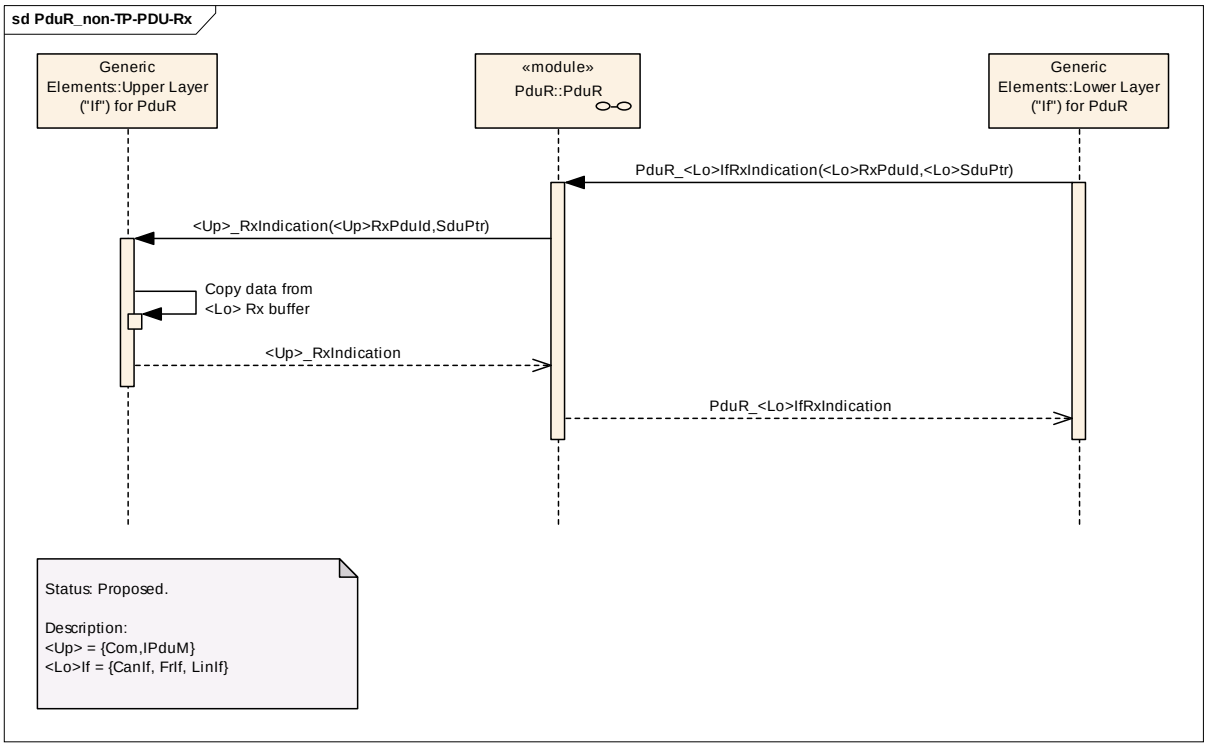

Figure 6: non TP-PDU-Rx

The reception of an I-PDU received from a transport protocol module (TP PDU Rx) is shown by Figure 7. The loop “TP Rx operation” is executed for each received N-PDU. Depending on the status of the receive buffer (undefined, full or enough space) a new receive buffer is requested. Then the data of the received N-PDU will be copied into the receive buffer. In case of an error or after the last N-PDU has been received an indication is provided.

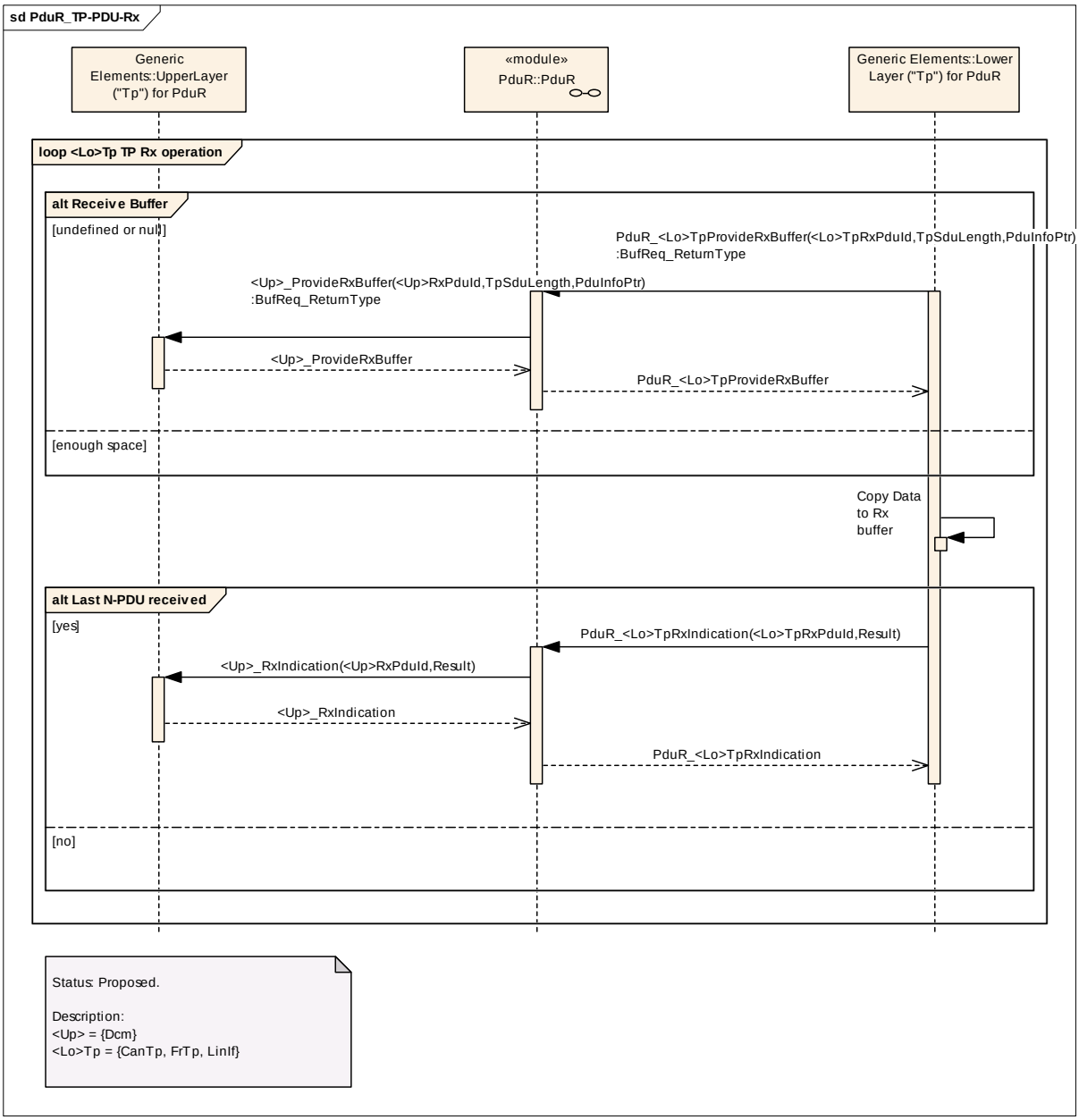

Figure 7: TP-PDU-Rx

9.3 PDU Transmission

The transmission of an I-PDU directly via an interface module (non-TP PDU Tx) is shown by Figure 8 (without trigger transmit) and Figure 9 (with trigger transmit). In the first case the data to be transmitted is provided via the PduInfoPtr parameter of

the transmit request. Therefore the data will be copied by the interface module and transmitted on the related bus. If statically configured for the PDU a transmit confirmation is provided.

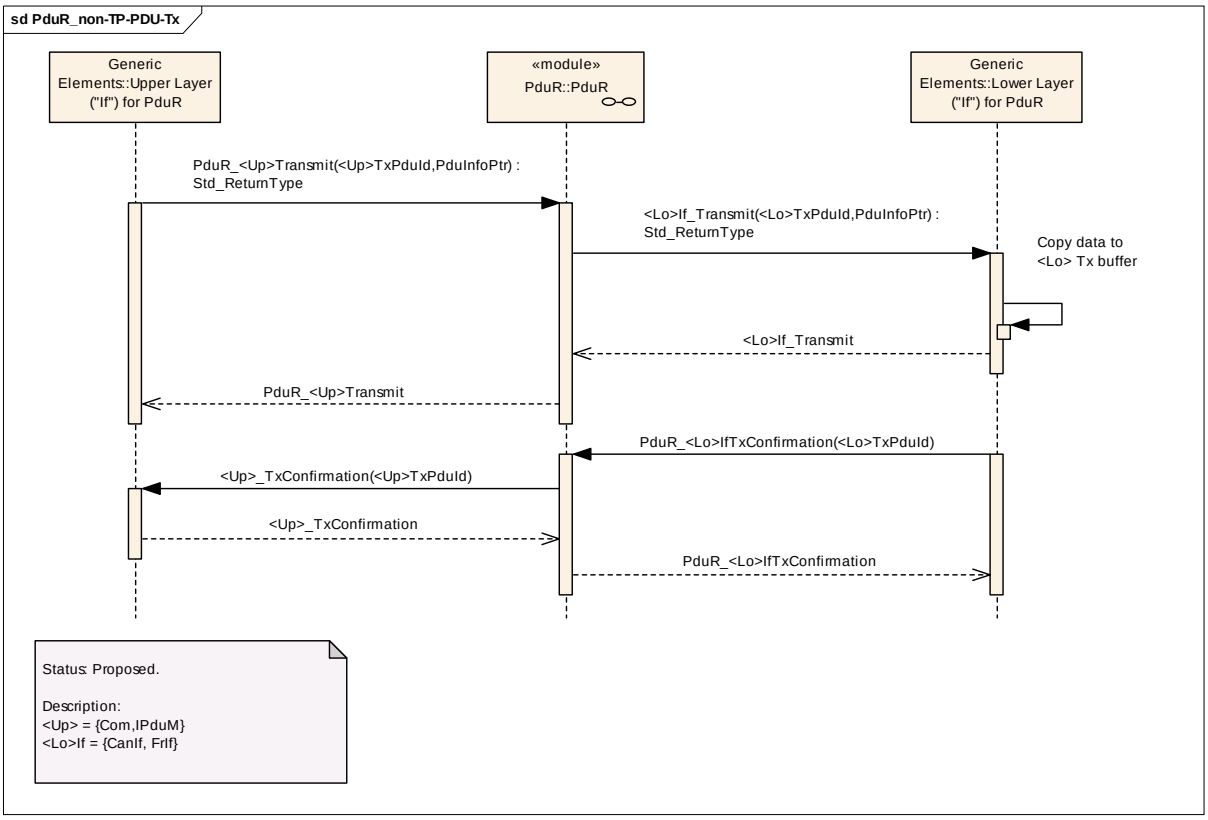

Figure 8: non TP-PDU-Tx without trigger transmit

In case a PDU is configured to use the TriggerTransmit data provision (Figure 9), the data will not be provided as part of the transmit request but will later be retrieved by the interface module via the function PduR_<Lo>IfTriggerTransmit which in turn will be forwarded by the PDU Router to the upper layer module by calling <Up>_TriggerTransmit. Here the data will be copied by the upper layer module. The interface module will transmit the data on the related bus and will provide a transmit confirmation if statically configured for the PDU.

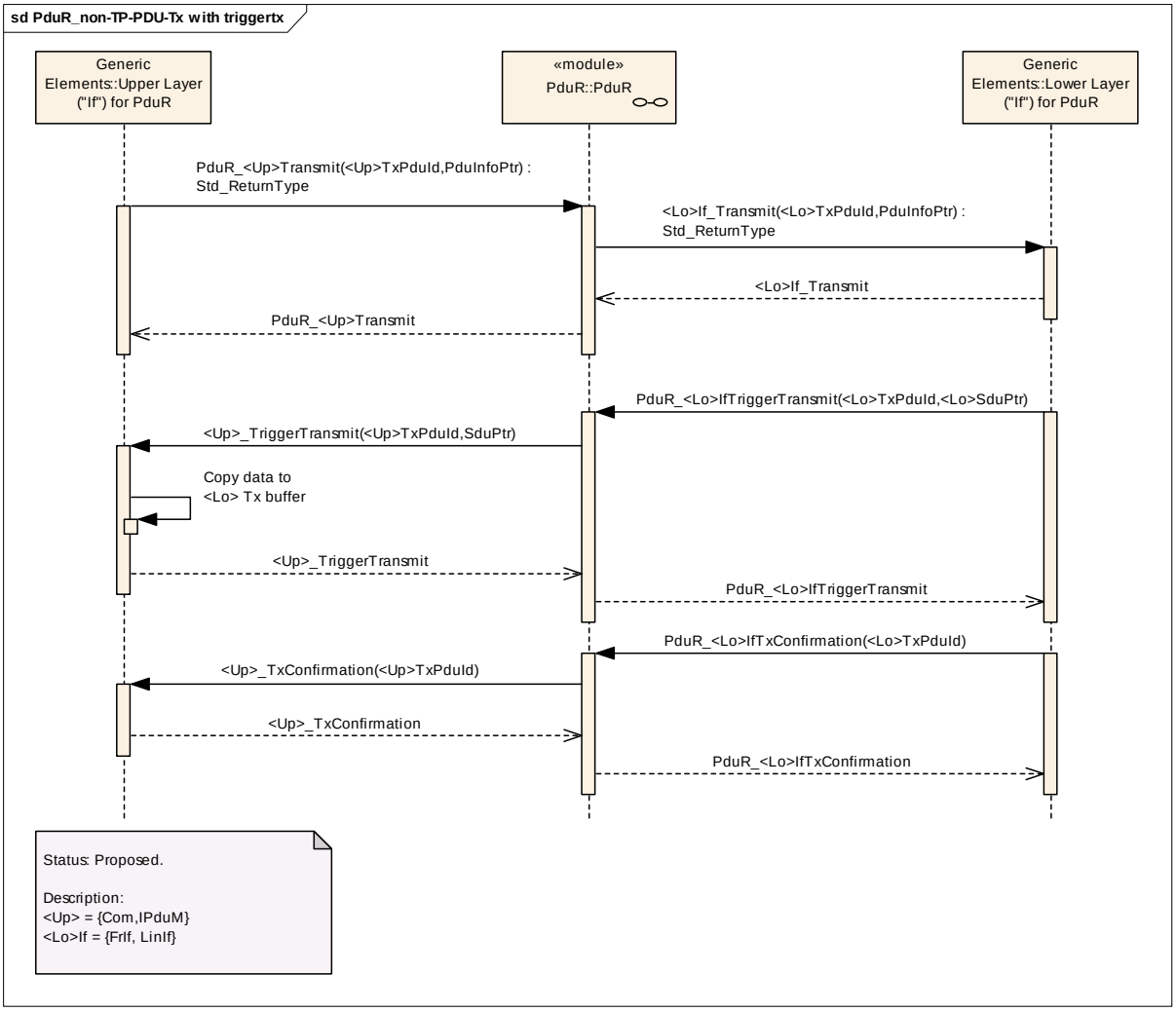

Figure 9: non-TP-PDU-Tx with trigger transmit

A multicast transmission via two interface modules (multicast non TP PDU Tx) is shown by Figure 10. The PDU to be transmitted via the second interface module is configured to use TriggerTransmit data provision and the PDU to be transmitted via the first interface module (rightmost line) is configured to use direct data provision. In case of multicasts no transmit confirmation will be provided.

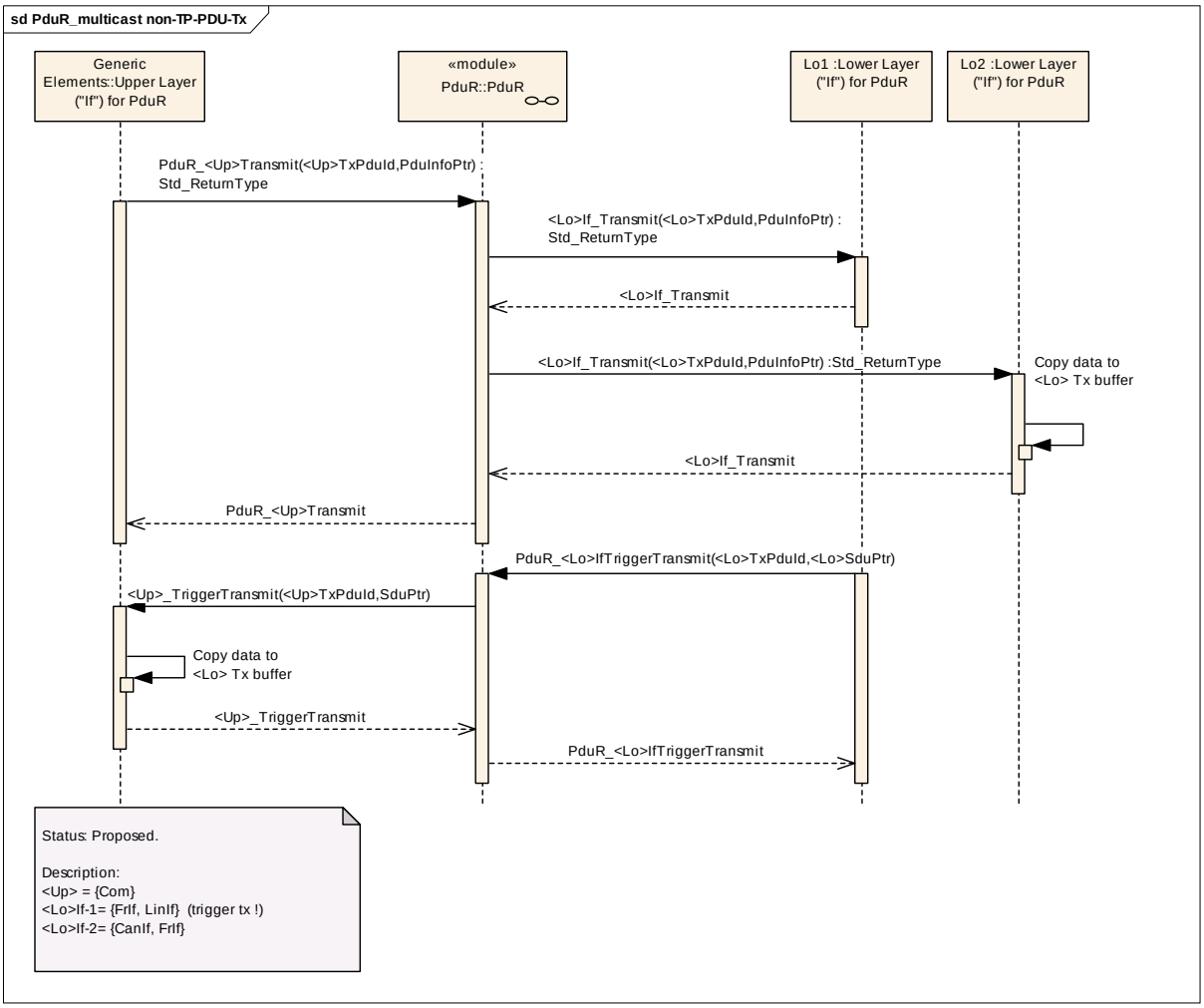

Figure 10: multicast non-TP PDU Tx

Figure 11 shows the transmission of an I-PDU via a transport protocol module (TP PDU Tx). First the transmit request is forwarded by the PDU Router to the related TP module. Then the TP module executes the loop “TP Tx operation” for each N-PDU transmission. Depending on the status of the transmit buffer (undefined or all data processed) a new transmit buffer is requested. The TP module will transmit an N-PDU by reading the data from the transmit buffer. For an efficient usage of the transmit buffer the buffer size should be a multiple of the N-PDU data length. In case of an error or after the last N-PDU has been transmitted a confirmation is provided.

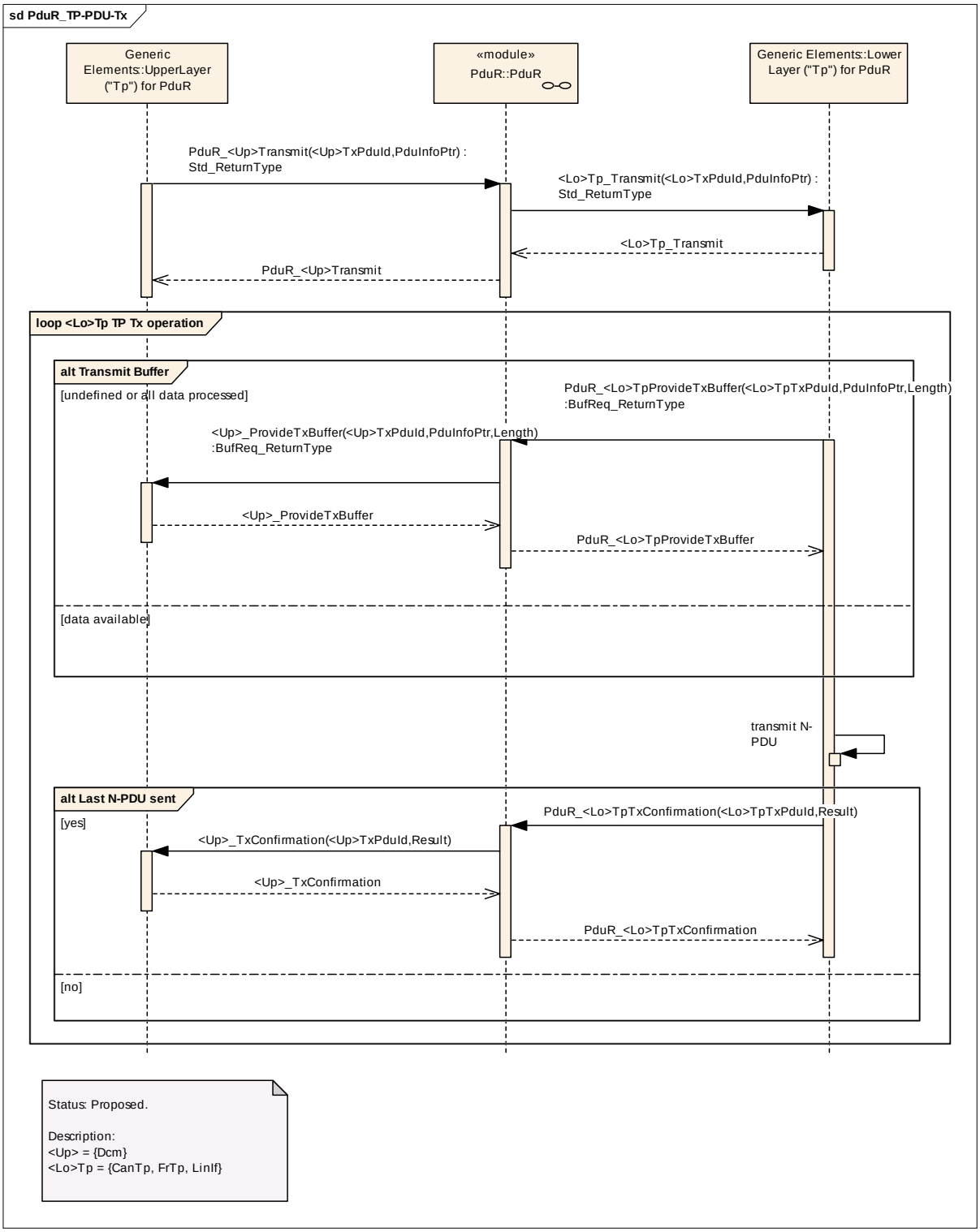

Figure 11: TP-PDU-Tx

A multicast single frame TP transmission via two TP modules (multicast SF TP PDU Tx) is shown by Figure 12. In contrast to a multiple frame TP transmission no loop operations have to be executed as only a single N-PDU will be transmitted (on each bus). <Up>_ProvideTxBuffer is only called when the PDU Router is requested to provide a transmit buffer by the first TP module. The buffer provided by the upper layer module will then be used for all TP transmissions and will be released after the

last TP module confirms transmission (<Up>_TxConfirmation will be called within the PduR_<Lo>TpTxConfirmation call of the last TP module).

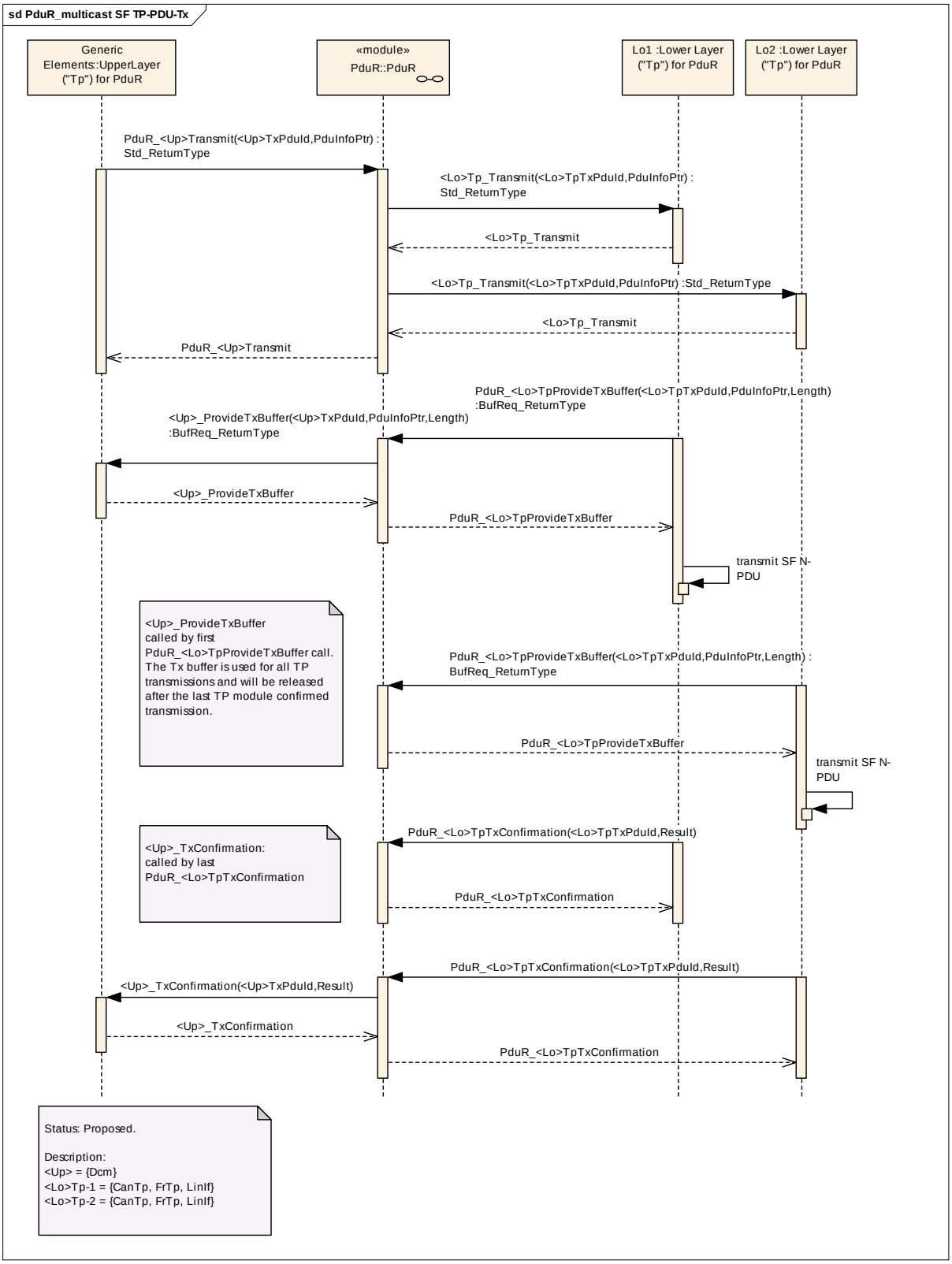

Figure 12: multicast SF TP-PDU-Tx

9.4 PDU Gateway

Figure 13 and Figure 14 show the PDU Router acting as a direct PDU gateway between two interface modules (non TP PDU Gateway without rate conversion). PDUs received from one bus (interface module 2, rightmost line) shall be forwarded to the other bus (interface module 1). First of all it is shown that no upper layer module is involved in the gateway operation (empty line, leftmost).

In the first case (Figure 13) the PDU to be transmitted via interface module 1 is configured to use direct data provision. Therefore the data pointer received from interface module 2 (rightmost line) will be provided via the PduInfoPtr parameter of the transmit request to interface module 2. The latter will directly copy the data from the receiving interface module and transmit it on the destination bus.

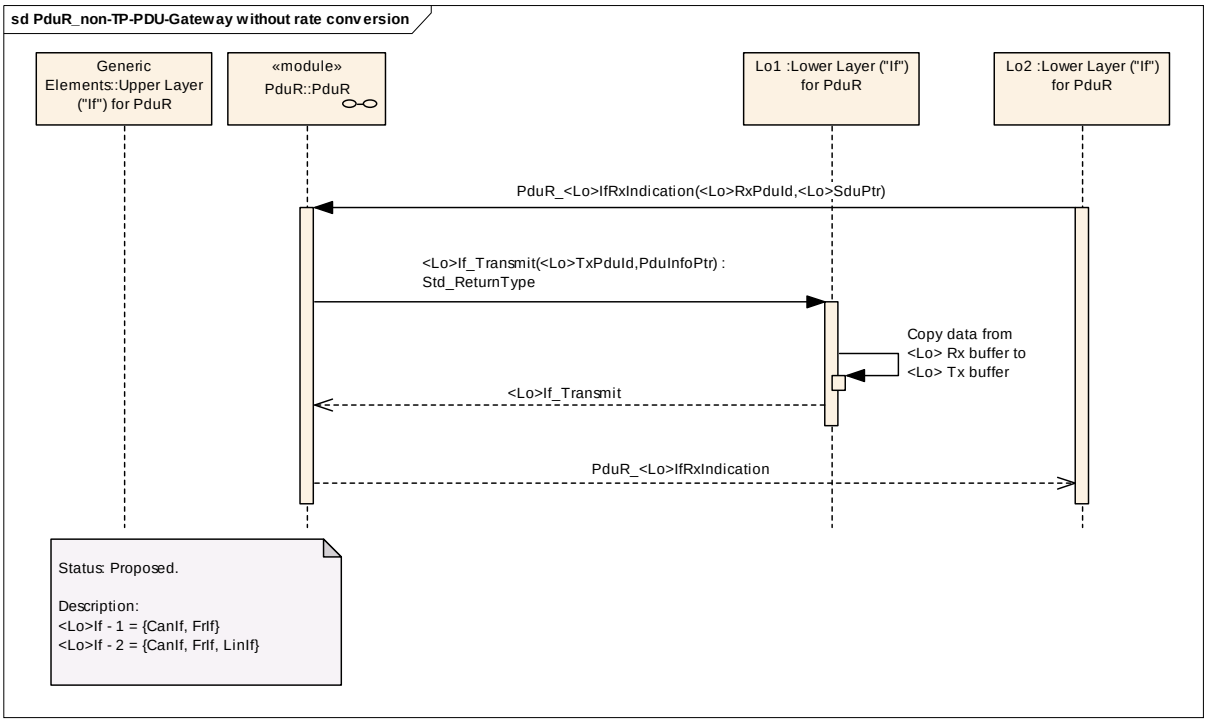

Figure 13: non-TP-PDU-Gateway without rate conversion

In the second case (Figure 14) the PDU to be transmitted via interface module 1 is configured to use TriggerTransmit data provision. Therefore the data received from interface module 2 will not be provided as part of the transmit request to interface module 1. It will later be retrieved by interface module 1 via the TriggerTransmit function. As TriggerTransmit is decoupled from the PduR_<Lo>IfRxIndication the PDU Router has to provide a dedicated PDU transmit buffer to store the received PDU. Later on when TriggerTransmit is called, the PDU Router copies the data from the PDU transmit buffer to a place requested by interface module 1 which will transmit it on the destination bus.

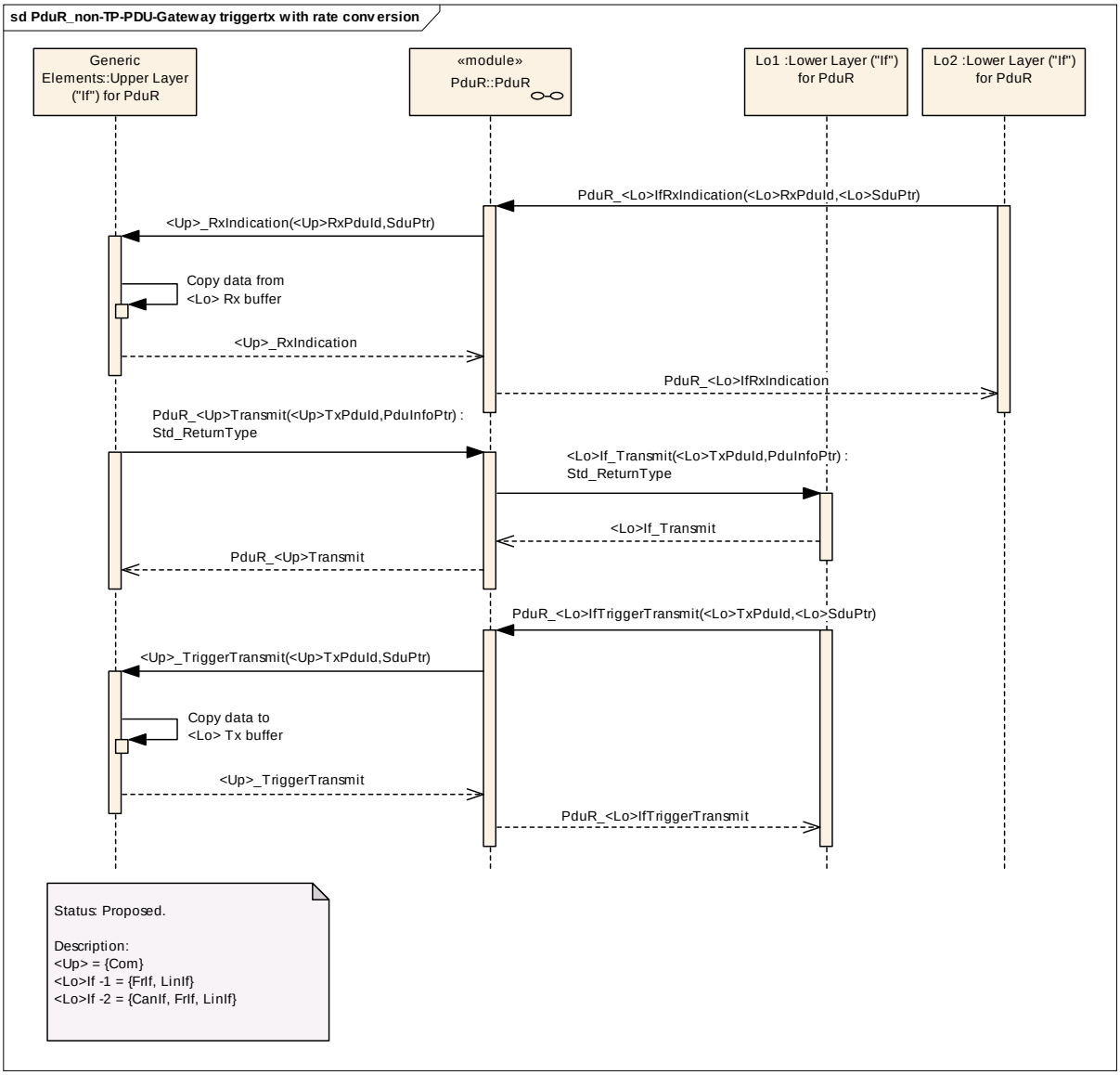

Figure 14: non-TP-PDU-Gateway without rate conversion (trigger transmit version)

A PDU Gateway between two interface modules with rate conversion is not directly supported by the PDU Router. But as shown by Figure 15 this could be done by AUTOSAR COM. It simply consists of two parts: (1) PDU reception from interface module 2 (cp. Figure 6) and (2) PDU transmission via interface module 1 (cp. Figure 9 for trigger transmit data provision - in case of direct data transmission the transmit part is according to Figure 8).

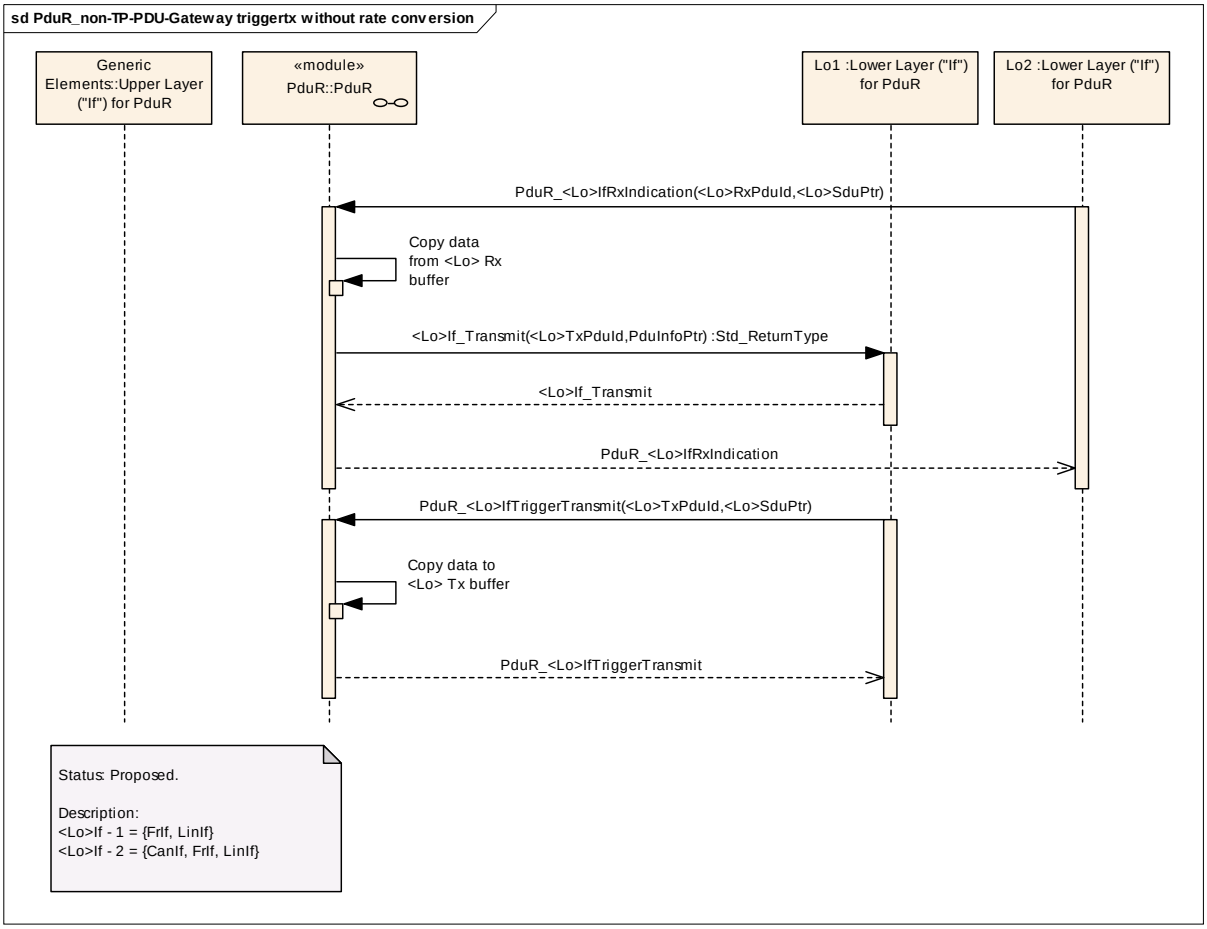

Figure 15: non-TP-PDU Gateway with rate conversion (trigger transmit version)

The sequence diagram of a PDU gateway between two TP modules (TP PDU Gateway) is shown by Figure 16 and Figure 17. Data received from one bus (TP module 1) shall be forwarded to another bus (TP module 2, rightmost line).

First of all it is shown that no upper layer module is involved in the gateway operation (empty line, leftmost). Basically the gateway consists of two parts: (1) TP PDU reception from TP module 1 (cp. Figure 7) and (2) TP PDU transmission via TP module 2 (cp. Figure 11). As the PDU Router shall support routing on-the-fly, the transmission via TP module 2 has to be started before the complete I-PDU is received via TP module 1. By each call of PduR_<Lo>TpProvideRxBuffer or PduR_<Lo>TpRxIndication the previously provided receive buffer is released and can be used as a transmit buffer for TP transmission on the destination bus. Hence the usage of a large buffer causes store-and-forward routing and the usage of small buffers causes on-the-fly routing. To start the TP transmission on the destination bus the PDU Router will call <Lo>Tp_Transmit when the first receive buffer is released by the receiving TP module (either within PduR_<Lo>TpProvideRxBuffer or within PduR_<Lo>TpRxIndication as shown by Figure 16).

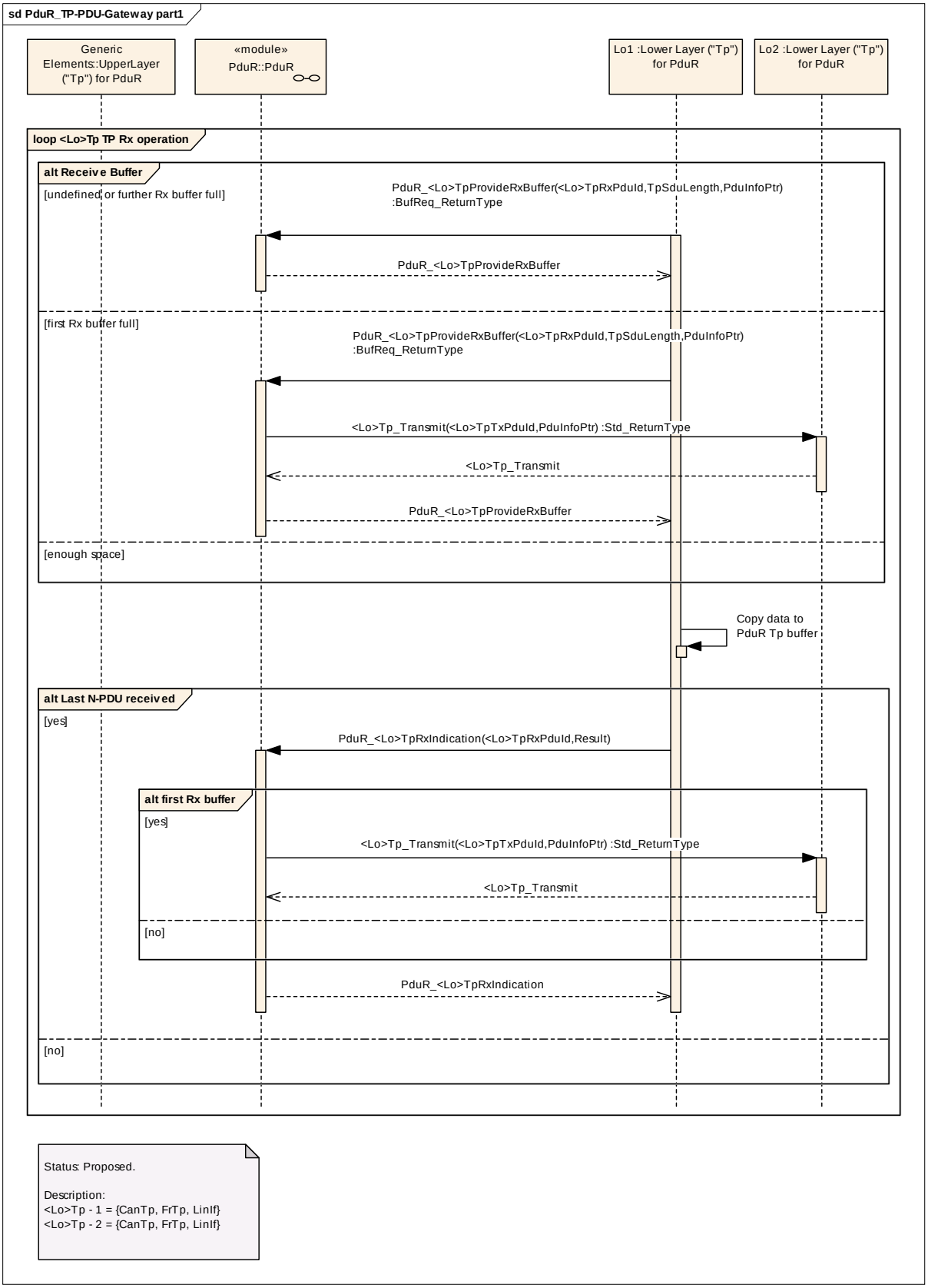

Figure 16: TP PDU Gateway part 1

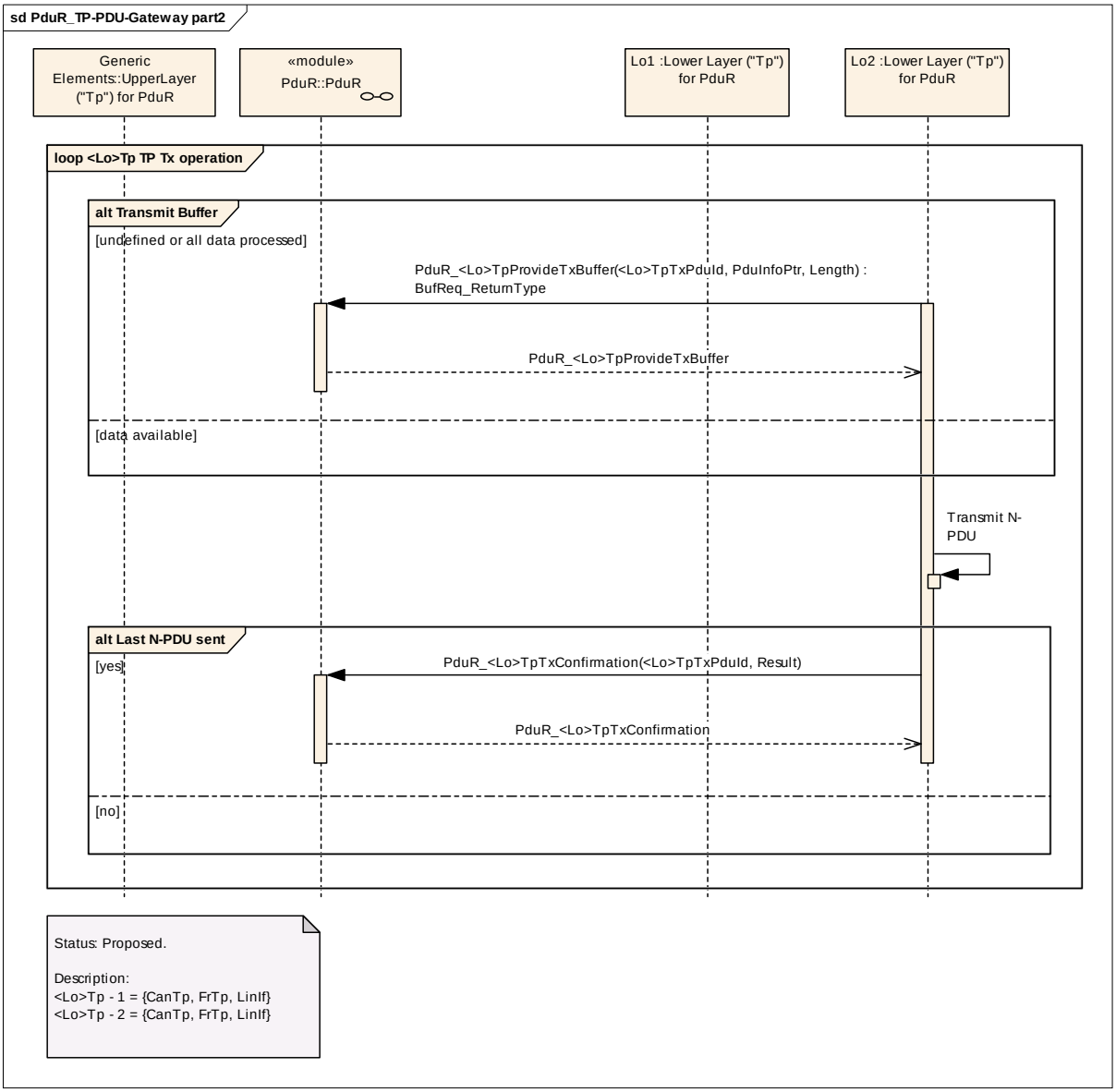

Figure 17: TP PDU Gateway part 2

Note: If the retry feature is configured for a TP transmission, the related TP module will request a buffer of length equal to the block size specified by the receiver on the destination bus. Therefore the buffer(s) used for TP reception on the source bus must be as large as the maximum block size used for the transmission on the destination bus or belong to a linear buffer of at least that size. If no retry is used the requested TP transmit buffer may be of arbitrary size and therefore no restrictions regarding the buffers used for TP reception apply.

Figure 18 shows a TP PDU Gateway with two destination busses (multicast SF TP PDU Gateway). Also for this gateway operation no upper layer module is involved (empty line, leftmost). In contrast to a multiple frame TP reception or transmission no loop operations have to be executed as only a single N-PDU will be received or transmitted (on each bus) respectively. The PDU Router provides the receive buffer when it is requested by the receiving TP module (TP module 1). Within PduR_<Lo>TpRxIndication the PDU Router will request a TP transmission at TP

module 2 and TP module 3. The released receive buffer will be provided as a transmit buffer to TP module 2 and TP module 3 when requested via PduR_<Lo>TpProvideTxBuffer. Then the single frame N-PDU will be transmitted on the destination busses. When the last TP module calls PduR_<Lo>TpTxConfirmation (TP module 3 as shown by Figure 18) the transmit buffer will be released.

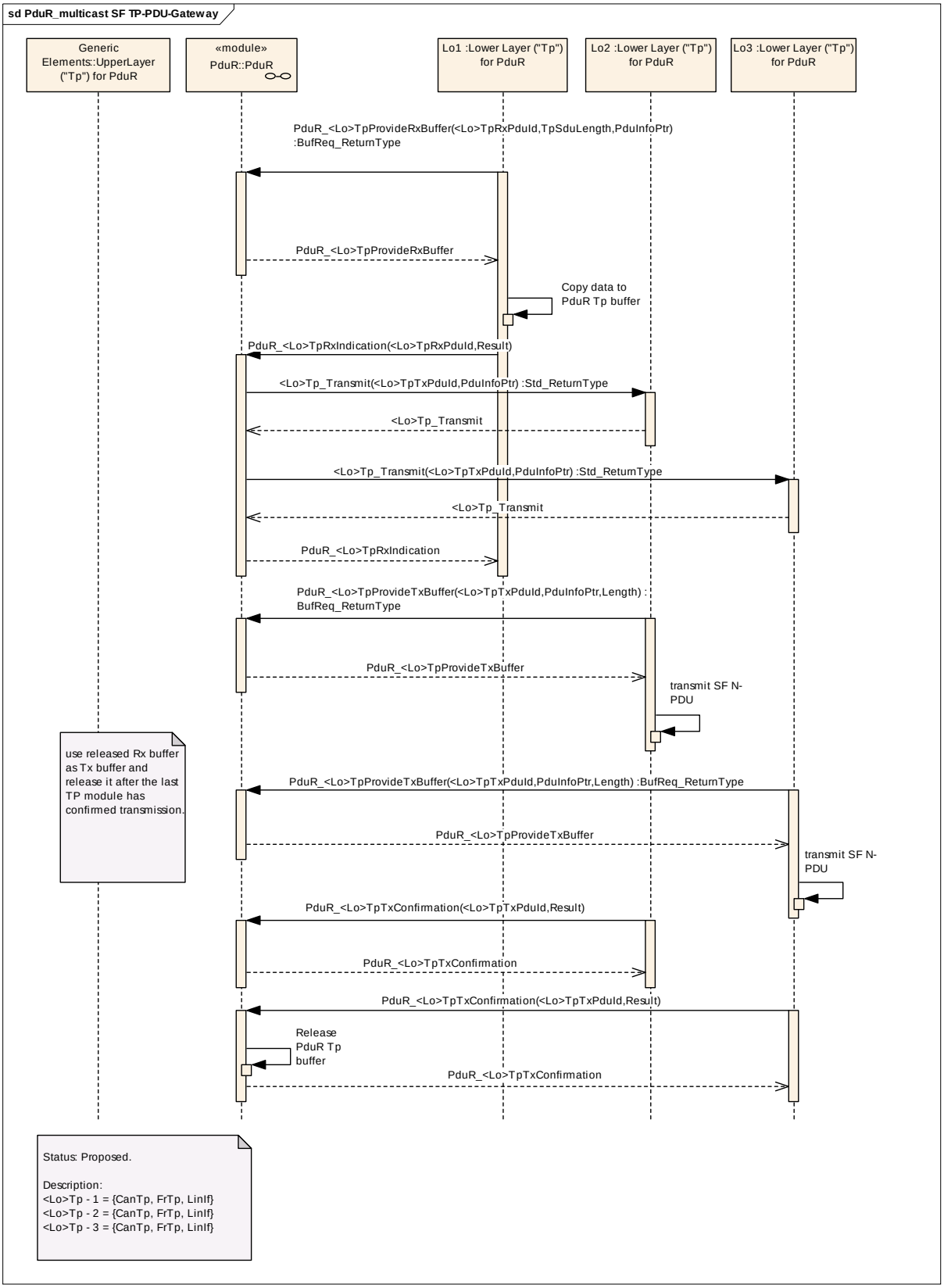

Figure 18: multicast SF TP PDU Gateway

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 PDU Router.

Chapter 10.3 specifies published information of the module PDU Router.

10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [15]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

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

10.1.1 Configuration and configuration parameters

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

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

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., Variant PreCompile: only pre-compile time configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

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

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

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

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

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

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

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

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

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

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in chapter 7 and chapter 8. An overview of the top-level PDU Router configuration container PduR is shown in Figure 19.

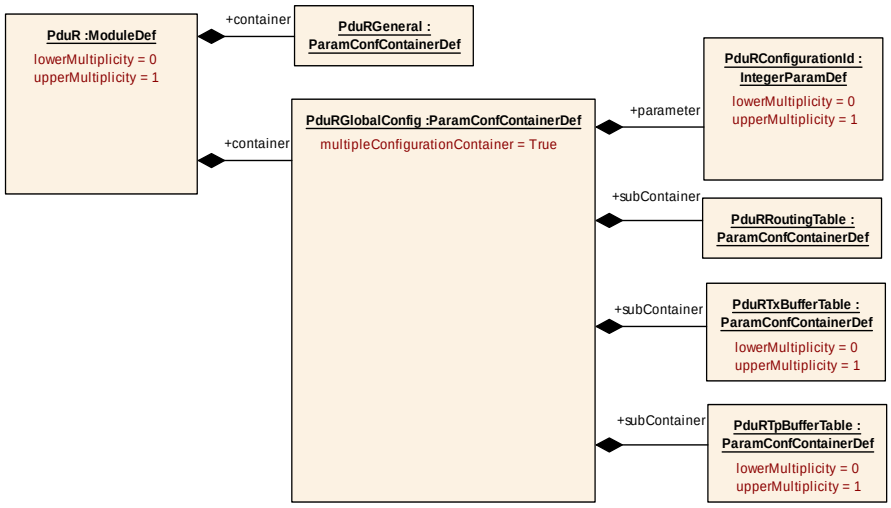

Figure 19: PDU Router Configuration Overview - PduR

Figure 20 provides an overview of the containers and configuration parameters that describe the PDU Router routing paths.

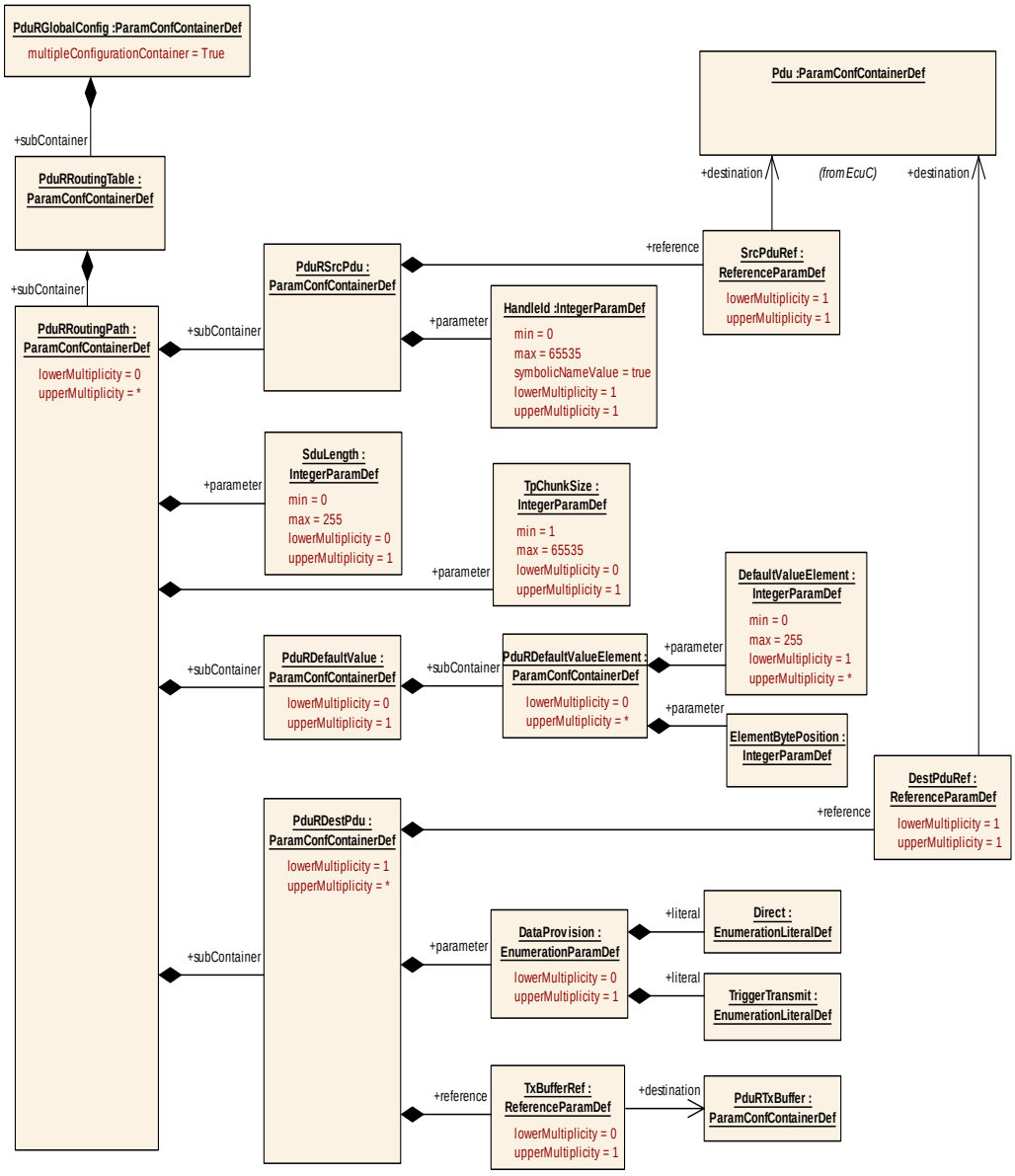

Figure 20: PDU Router Configuration Overview – Routing Paths

10.2.1 Variants

There are three configuration parameter sets defined for the PDU Router. If the configuration class of a configuration parameter is the same for all configuration parameter sets, the term “all Variants” is used instead of listing all possible variants.

PDUR425: VARIANT-PRE-COMPILE: The configuration parameter set for the PDU Router module contains only pre-compile time configuration parameters.

This variant is only possible in zero-cost operation.

PDUR427: VARIANT-POST-BUILD: The configuration parameter set for the PDU Router module contains a mix of pre-compile time, link time and post-build time configuration parameters.

10.2.2 PduR

Module Name	**PduR**
Module Description	Configuration of the PduR (PDU Router) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRGeneral	1	This container is a subcontainer of PduR and specifies the general configuration parameters of the PDU Router.
PduRGlobalConfig	1	This container contains the global configuration parameter of the PduR. It is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set.

10.2.3 PduRGeneral

SWS Item	**--**
Container Name	PduRGeneral
Description	This container is a subcontainer of PduR and specifies the general configuration parameters of the PDU Router.
Configuration Parameters	

SWS Item	**--**		
Name	PduRCanIfSupport {PDUR_CANIF_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for CAN interface.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRCanTpSupport {PDUR_CANTP_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for CAN TP.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**
Name	PduRComSupport {PDUR_COM_SUPPORT}
Description	Configuration parameter to enable or disable PDU Router support for COM.
Multiplicity	1
Type	BooleanParamDef
Default value	--

ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRDcmSupport {PDUR_DCM_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for DCM.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRDevErrorDetect {PDUR_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRFifoTxBufferSupport {PDUR_FIFO_TX_BUFFER_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for FIFOs as PDU transmit buffers; if PDUR_GATEWAY_OPERATION is disabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRFrIfSupport {PDUR_FRIF_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for FlexRay interface.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**

Name	PduRFrTpSupport {PDUR_FRTP_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for FlexRay TP.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRGatewayOperation {PDUR_GATEWAY_OPERATION}		
Description	Configuration parameter to enable or disable PDU Router gateway operation; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRIPduMSupport {PDUR_IPDUM_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for IPDUM; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRLinIfSupport {PDUR_LINIF_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for LIN interface.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRLinTpSupport {PDUR_LINTP_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for LIN TP.		
Multiplicity	1		

Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMemorySize {PDUR_MEMORY_SIZE}		
Description	Memory size reserved for PDU Router buffers. Only required for gateway operation.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMinimumRoutingLoModule {PDUR_MINIMUM_ROUTING_LO_MODULE}		
Description	Lower layer module to be used for minimum routing; this parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	CAN_IF	Can Interface	
	CAN_TP	Can TP	
	FR_IF	FlexRay Interface	
	FR_TP	FlexRay TP	
	LIN_IF	Lin Interface	
	LIN_TP	Lin TP	
ConfigurationClass	***Pre-compile time***	--	
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMinimumRoutingLoRxPduId {PDUR_MINIMUM_ROUTING_LO_RXPDUID}		
Description	Receive PDU identifier of the lower layer module which shall be used at the PDU Router interface to the lower layer module specified by PDUR_MINIMUM_ROUTING_LO_MODULE for minimum routing; this parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMinimumRoutingLoTxPduId {PDUR_MINIMUM_ROUTING_LO_TXPDUID}		
Description	Transmit PDU identifier of the lower layer module which shall be used at the PDU Router interface to the lower layer module specified by PDUR_MINIMUM_ROUTING_LO_MODULE for minimum routing; this parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMinimumRoutingUpModule {PDUR_MINIMUM_ROUTING_UP_MODULE}		
Description	Upper layer module to be used for minimum routing; this parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	COM	COM	
	DCM	DCM	
ConfigurationClass	***Pre-compile time***	--	
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMinimumRoutingUpRxPduId {PDUR_MINIMUM_ROUTING_UP_RXPDUID}		
Description	Receive PDU identifier of the upper layer module which shall be used at the PDU Router interface to the upper layer module specified by PDUR_MINIMUM_ROUTING_UP_MODULE for minimum routing; this parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**
Name	PduRMinimumRoutingUpTxPduId {PDUR_MINIMUM_ROUTING_UP_TXPDUID}
Description	Transmit PDU identifier of the upper layer module which shall be used at the PDU Router interface to the upper layer module specified by PDUR_MINIMUM_ROUTING_UP_MODULE for minimum routing; this

	parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMulticastFromIfSupport {PDUR_MULTICAST_FROMIF_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for multicasts from an interface module to upper layer modules or lower layer interface modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMulticastFromTpSupport {PDUR_MULTICAST_FROMTP_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for multicasts from a TP module to upper layer modules or lower layer TP modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRMulticastToIfSupport {PDUR_MULTICAST_TOIF_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for multicasts from an upper layer module to interface modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**

Name	PduRMulticastToTpSupport {PDUR_MULTICAST_TOTP_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for multicasts from an upper layer module to TP modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRSbTxBufferSupport {PDUR_SB_TX_BUFFER_SUPPORT}		
Description	Configuration parameter to enable or disable PDU Router support for single buffers as PDU transmit buffers; if PDUR_GATEWAY_OPERATION is disabled, this parameter has to be disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRSingleIf {PDUR_SINGLE_IF}		
Description	Single interface module in case zero cost operation is enabled (PDUR_ZERO_COST_OPERATION).		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	CAN_IF	Can interface	
	FR_IF	FlexRay interface	
	LIN_IF	Lin interface	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRSingleTp {PDUR_SINGLE_TP}		
Description	Single transport protocol module in case zero cost operation is enabled (PDUR_ZERO_COST_OPERATION).		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	CAN_TP	Can TP	
	FR_TP	FlexRay TP	
	LIN_TP	Lin TP	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRVersionInfoApi {PDUR_VERSION_INFO_API}		
Description	Activates/Deactivates the Version Info API.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**--**		
Name	PduRZeroCostOperation {PDUR_ZERO_COST_OPERATION}		
Description	All routing paths are implicitly defined and the communication modules directly above or below the PDU Router shall directly call each other without using PDU Router functions (zero cost operation). The configuration parameters PDUR_SINGLE_IF and PDUR_SINGLE_TP are used to specify the related lower layer module.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.4 PduRTxBufferTable

SWS Item	**PDUR243 :**
Container Name	PduRTxBufferTable
Description	This container is a subcontainer of PduR and contains the definition of all transmit buffers (used by specific non-TP PDUs; only required for PDU Router gateway operation). This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_GATEWAY_OPERATION is enabled.
Configuration Parameters	

SWS Item	**--**		
Name	PduRMaxTxBufferNumber {PDUR_MAX_TX_BUFFER_NUMBER}		
Description	maximum number of transmit buffers		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-POST-BUILD
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRTxBuffer	0..*	This container is a subcontainer of PduRTxBufferTable and

		specifies a transmit buffer for a non-TP PDU.

10.2.5 PduRTxBuffer

SWS Item	**PDUR244 :**
Container Name	PduRTxBuffer
Description	This container is a subcontainer of PduRTxBufferTable and specifies a transmit buffer for a non-TP PDU.
Configuration Parameters	

SWS Item	**--**		
Name	Depth		
Description	Specifies the depth of the buffer		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	Length		
Description	Length of the buffer.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.6 PduRRoutingTable

SWS Item	**PDUR247 :**
Container Name	PduRRoutingTable
Description	PDU Router routing table is a subcontainer ofPduR. This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_ZERO_COST_OPERATION is disabled.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRRoutingPath	0..*	This container is a subcontainer of PduRRoutingTable and specifies the routing path of a PDU.

10.2.7 PduRRoutingPath

SWS Item	**PDUR248 :**

Container Name	PduRRoutingPath
Description	This container is a subcontainer of PduRRoutingTable and specifies the routing path of a PDU.
Configuration Parameters	

SWS Item	**--**		
Name	SduLength		
Description	Length of PDU data (SDU). Only required if a TX buffer is configured.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	TpChunkSize		
Description	Chunk size for routing on the fly. Defines the number of bytes which shall be received before transmission on the destination bus may start. Only required for TP gateway PDUs. The TpChunkSize shall not be larger than the length of the related TP Buffer.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRDefaultValue	0..1	This container is a subcontainer of PduRRoutingPath and specifies the default value of the I-PDU. Only required for gateway operation and if at least one PDU specified by PduRDestPdu uses TriggerTransmit Data provision. Represented as an array of IntegerParamDef.
PduRDestPdu	1..*	This container is a subcontainer of PduRRoutingPath and specifies one destination for the PDU to be routed.
PduRSrcPdu	1	This container is a subcontainer of PduRRoutingPath and specifies the source of the PDU to be routed.

10.2.8 PduRSrcPdu

SWS Item	**PDUR288 :**
Container Name	PduRSrcPdu
Description	This container is a subcontainer of PduRRoutingPath and specifies the source of the PDU to be routed.
Configuration Parameters	

SWS Item	**--**
Name	HandleId
Description	PDU identifier assigned by PDU Router.

Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	SrcPduRef		
Description	Source PDU reference; reference to unique PDU identifier which shall be used for the requested PDU Router operation.		
Multiplicity	1		
Type	Reference to Pdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.9 PduRDestPdu

SWS Item	**PDUR249 :**
Container Name	PduRDestPdu
Description	This container is a subcontainer of PduRRoutingPath and specifies one destination for the PDU to be routed.
Configuration Parameters	

SWS Item	**--**		
Name	DataProvision		
Description	Specifies how data are provided: direct (as part of the Transmit call) or via the TriggerTransmit callback function. Only required for non-TP gateway PDUs.		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	DIRECT	direct data provision	
	TRIGGER_TRANSMIT	trigger transmit data provision	
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	DestPduRef		
Description	Destination PDU reference; reference to unique PDU identifier which shall be used by the PDU Router instead of the source PDU ID when calling the related function of the destination module.		
Multiplicity	1		
Type	Reference to Pdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**		
Name	TxBufferRef		
Description	Specifies the assigned transmit buffer. Only required for specific non-TP gateway PDUs.		
Multiplicity	0..1		
Type	Reference to PduRTxBuffer		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.10 PduRDefaultValue

SWS Item	**--**
Container Name	PduRDefaultValue
Description	This container is a subcontainer of PduRRoutingPath and specifies the default value of the I-PDU. Only required for gateway operation and if at least one PDU specified by PduRDestPdu uses TriggerTransmit Data provision. Represented as an array of IntegerParamDef.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRDefaultValueElement	0..*	Each value element is represented by the element and the position in an array.

10.2.11 PduRDefaultValueElement

SWS Item	**--**
Container Name	PduRDefaultValueElement
Description	Each value element is represented by the element and the position in an array.
Configuration Parameters	

SWS Item	**--**		
Name	DefaultValueElement		
Description	The default value consists of a number of elements. Each element is one byte long and the number of elements is specified by SduLength. The position of this parameter in the container is specified by the ElementBytePosition parameter.		
Multiplicity	1..*		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**--**
Name	ElementBytePosition

Description	This parameter specifies the byte position of the element within the default value		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.12 PduRTpBufferTable

SWS Item	**PDUR245 :**
Container Name	PduRTpBufferTable{TpBufferTable}
Description	This container is a subcontainer of PduR and contains the definition of all TP buffers (only required for PDU Router gateway operation). This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_GATEWAY_OPERATION is enabled.
Configuration Parameters	

SWS Item	**--**		
Name	PduRMaxTpBufferNumber {PDUR_MAX_TP_BUFFER_NUMBER}		
Description	maximum number of TP buffers.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-POST-BUILD
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRTpBuffer	0..*	This container is a subcontainer of PduRTpBufferTable and specifies a TP buffer.

10.2.13 PduRTpBuffer

SWS Item	**PDUR246 :**
Container Name	PduRTpBuffer
Description	This container is a subcontainer of PduRTpBufferTable and specifies a TP buffer.
Configuration Parameters	

SWS Item	**--**	
Name	Length	
Description	Length of the buffer.	
Multiplicity	1	
Type	IntegerParamDef	
Range	1 .. 65535	
Default value	--	

ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.14 PduRGlobalConfig

SWS Item	**--**
Container Name	PduRGlobalConfig [Multi Config Container]
Description	This container contains the global configuration parameter of the PduR. It is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set.
Configuration Parameters	

SWS Item	**--**		
Name	PduRConfigurationId {PDUR_CONFIGURATION_ID}		
Description	unique configuration identifier of post-build time configuration; this parameter shall be used if PDUR_ZERO_COST_OPERATION is disabled; otherwise it shall not be used.		
Multiplicity	0..1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRRoutingTable	1	PDU Router routing table is a subcontainer ofPduR. This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_ZERO_COST_OPERATION is disabled.
PduRTpBufferTable	0..1	This container is a subcontainer of PduR and contains the definition of all TP buffers (only required for PDU Router gateway operation). This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_GATEWAY_OPERATION is enabled.
PduRTxBufferTable	0..1	This container is a subcontainer of PduR and contains the definition of all transmit buffers (used by specific non-TP PDUs; only required for PDU Router gateway operation). This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_GATEWAY_OPERATION is enabled.

10.3 Published Information

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

The standard common published information like

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

is provided in the BSW Module Description Template (see [18] Figure 4.1 and Figure 7.1).

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

10.4 Plausibility checks of configuration

PDUR225: During system generation the ECU configuration tool shall perform plausibility checks according to the following rules and constraints:

(1) Sum of memory size used for all PDU transmit buffer (sum of each TxBuffer length) plus the memory reserved for TP buffers of the PDU Router module (sum of each TpBuffer length) must not exceed the reserved memory for PDU Router module buffers specified by the pre-compile-time configuration parameter PduRMemorySize.

(2) If the pre-compile time configuration parameter PduRZeroCostOperation is enabled all conditions defined must be fulfilled (e.g. PduRCanIfSupport and PduRFrIfSupport must not be enabled at the same time, PduRGatewayOperation must not be enabled, ...).

(3) If PduRGatewayOperation is disabled, the pre-compile time configuration parameters PduRSbTxBufferSupport, PduRFifoTxBufferSupport, PduRMulticastFromIfSupport and PduRMulticastFromTpSupport must be disabled as well.
(4) TpChunkSize (PduRRoutingTable/PduRRoutingPath, see 10.2.6) shall not be greater than the length of the largest TP Buffer (PduRTpBufferTable/PduRTpBuffer, see10.2.12).

10.5 Example structure of Routing tables

This chapter shows example structures of routing tables that contain the properties of each PDU. It does not specify the internals of the PDU Router but shall rather serve as example for better understanding of API and PDU name spaces. The IPDUM is not considered by these examples.

Note: The first row of those tables contain the structure element name and the first column the array index number of the element. If not all routing capabilities are required, some of the tables or parts of the tables may be omitted. For a better readability the tables shown below are not fully optimized.

10.5.1 Routing tables for communication via interface modules

Routing table used by PduR_ComTransmit for IfTxPDUs (transmitted by COM):

ComTxPduId	*TargetFctPtr*	*TargetPduId*
0	CanIf_Tansmit	0
1	FrIf_Transmit	0
2	CanIf_Tansmit	1
3	MultiIf_Transmit	0
4	MultiIf_Transmit	2
…	…	…

The first three entries represent normal PDU transmit operations from Com via CanIf or FrIf respectively, the remaining two entries are related to multicast PDU transmit operations from Com via FrIf and CanIf. For the latter an internal PDU Router function (MultiIf_Transmit) and an additional routing table is used.

Routing table used by MultiIf_Transmit for IfTxPDUs:

Index	*MPduId*	*TargetFctPtr*	*TargetPduId*
0	0	FrIf_Transmit	2
1	0	CanIf_Transmit	3
2	2	CanIf_Transmit	4
3	2	FrIf_Transmit	3
4	4	NULL	0

The routing table for multicast PDU transmit operations contains multiple entries for each multicast PDU transmit request which is represented by MPduId. For a direct access to the related table entries the index value of the first PDU transmit request of a multicast operation is used as MPduId (e.g. 0 and 2). All subsequent entries with the same MPduId belong to the same multicast request. The execution of a multicast operation ends at an entry with a different MPduId.

Routing table used by PduR_<Lo>IfTxConfirmation and PduR_<Lo>IfTriggerTransmit for IfTxPDUs of <Lo>If:

CanIfTxPduId	*TargetFctPtr1*	*TargetPduId*
0	Com_TxConfirmation	0
1	Com_TxConfirmation	2
2	NULL	0
3	NULL	0
4	NULL	0
5	NULL	0
6	NULL	0
…	…	…

FrIfTxPduId	*TargetFctPtr1*	*TargetFctPrt2*	*TargetPduId*
0	Com_TriggerTransmit	Com_TxConfirmation	1
1	NULL	NULL	0
2	Com_TriggerTransmit	NULL	3
3	Com_TriggerTransmit	NULL	4
4	MG_IfTriggerTransmit	NULL	0
5	MG_IfTriggerTransmit	NULL	3
…	…	…	…

Not all <Lo>IfTxPduIds are used by the PDU Router; e.g. FrIfTxPduId = 1 may be used by FrNM (FlexRay Network Management module) or FrTp (FlexRay Transport Protocol module). If no transmit confirmation is configured, TargetFctPtr2 will be NULL; e.g. there is no a transmit confirmation for multicasts (CanIfTxPduId = 3 and 4, FrIfTxPduId = 2 and 3) or gateway operation (FrIfTxPduId = 4 and 5).

Routing table used by PduR_<Lo>IfRxIndication for IfRxPDUs received from <Lo>If:

CanIfRxPduId	*TargetFctPtr1*	*TargetPduId*
0	Com_RxIndication	0
1	MG_IfRxIndication	0
2	MG_IfRxIndication	1
...	...	...

FrIfRxPduId	*TargetFctPtr1*	*TargetPduId*
0	MG_IfRxIndication	4
1	Com_RxIndication	2
...	...	...

Routing table used by MG_IfRxIndication and MG_IfTriggerTransmit (functions for multicast and gateway operation) for IfRxPDUs and IfTxPDUs respectively:

Index	*MG PduId*	*TargetFct Ptr1*	*TargetFct Ptr2*	*Target PduId*	*SDU length*	*Buffer Type*	*TxBuffer Idx*		
0	0	NULL	FrIf_Transmit	4	8	1	0		G
1	1	NULL	CanIf_Transmit	5	8	0	0	M	G
2	1	Com_ RxIndicati on	NULL	1	8	0	0	M	
3	1	NULL	FrIf_Transmit	5	8	2	1	M	G
4	4	NULL	CanIf_Transmit	6	8	0	0		G
5	5	NULL	NULL	0	0	0	0		

SDU length:
0 ... undefined
>0 ... SDU length in bytes

BufferType:
0 ... no buffer (TxBufferIdx is not used)
1 ... single buffer
2 ... TT-FIFO buffer
3 ... D-FIFO buffer

TxBufferIdx … PDU transmit buffer index

The routing table shown above is used for gateway operation (G) and handling of multiple “receivers” (M). (The M/G markers are not part of the routing table.) Entries which belong to the same multicast/gateway operation are represented by the same MGPduId. For a direct access to the related table entries the index value of the first PDU receive or PDU transmit request of a multicast/gateway operation is used as MGPduId (e.g. 0, 1, and 4).

10.5.2 Routing tables for communication via transport protocol modules

Routing table used by PduR_DcmTransmit for TpTxPDUs (transmitted by DCM):

DcmTxPduId	*TargetFctPtr*	*TargetPduId*
0	CanTp_Transmit	0
1	CanTp_Transmit	1
2	FrTp_Transmit	0
3	MultiTp_Transmit	0
…	…	…

Routing table used by MultiTp_Transmit for TpTxPDUs:

Index	*MPduId*	*TargetFctPtr*	*TargetPduId*
0	0	FrTp_Transmit	1
1	0	CanTp_Transmit	2
2	2	NULL	0

Routing table used by PduR_<Lo>TpTxConfirmation and PduR_<Lo>TpProvideTx-Buffer for TpTxPDUs of <Lo>Tp:

CanTpTxPduId	*TargetFctPtr1*	*TargetFctPrt2*	*Target PduId*	*MultiTp*
0	Dcm_ProvideTxBuffer	Dcm_TxConfirmation	0	FALSE
1	Dcm_ProvideTxBuffer	Dcm_TxConfirmation	1	FALSE
2	Dcm_ProvideTxBuffer	Dcm_TxConfirmation	3	TRUE
3	MG_TpProvideTxBuffer	MG_TpTxConfirmation	1	TRUE
…	…	…	…	…

FrTpTxPduId	*TargetFctPtr1*	*TargetFctPrt2*	*Target PduId*	*MultiTp*
0	Dcm_ProvideTxBuffer	Dcm_TxConfirmation	2	FALSE
1	Dcm_ProvideTxBuffer	Dcm_TxConfirmation	3	TRUE
2	MG_TpProvideTxBuffer	MG_TpTxConfirmation	0	FALSE
3	MG_TpProvideTxBuffer	MG_TpTxConfirmation	3	TRUE
…	…	…	…	…

The column “MultiTp” indicates whether a condition for calling the configured target function applies or not. E.g. the third entry of the first table (CanTpTxPduId = 2) and the second entry of the second table (FrTPTxPduId = 1) belong to a multicast SF TP-PDU transmission; the target function Dcm_ProvideTxBuffer shall only be called at the first PduR_<Lo>TpProvideTxBuffer request and Dcm_TxConfirmation shall only be called at the last PduR_<Lo>TpTxConfirmation indication (see Figure 12).

Routing table used by PduR_<Lo>TpProvideRxBuffer or PduR_<Lo>TpRxIndication for TpRxPDUs received from <Lo>Tp:

CanTpRxPduId	*TargetFctPtr1*	*TargetFctPtr2*	*TargetPduId*
0	Dcm_ProvideRxBuffer	Dcm_RxIndication	0
...	...		...

FrTpRxPduId	*TargetFctPtr1*	*TargetFctPtr2*	*TargetPduId*
0	Dcm_ProvideRxBuffer	Dcm_RxIndication	1
1	MG_TpProvideRxBuffer	MG_TpRxIndication	0
2	MG_TpProvideRxBuffer	MG_TpRxIndication	1
3	Dcm_ProvideRxBuffer	Dcm_RxIndication	3
...	...		...

Routing table used by MG_TpProvideRxBuffer, MG_TpRxIndication and MG_TpProvideTxBuffer, MG_TpTxConfirmation (functions for multicast and gateway operation) for TpRxPDUs and TpTxPDUs respectively:

Index	*MG PduId*	*TargetFctPtr1*	*TargetFct Ptr2*	*TargetFct Ptr3*	*Target PduId*		
0	0	NULL	NULL	FrTp_Trans mit	2		G
1	1	NULL	NULL	CanTp_Tra nsmit	3	M	G
2	1	Dcm_ProvideRx Buffer	Dcm_RxInd ication	NULL	2	M	
3	1	NULL	NULL	FrTp_Trans mit	3	M	G
4	4	NULL	NULL	NULL	0		

M … Multicast, G … Gateway (The M/G markers are not part of the routing table)

11 Changes to Release 2

11.1 Deleted SWS Items

SWS Item	*Rationale*
PDUR321	Result of Bug#18912
PDUR322	Result of Bug#18912
PDUR323	Result of Bug#18912
PDUR165	Result of Bug#18912
PDUR426	Variant 2 removed as per Bug#20777

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
PDUR291	PDUR425, PDUR426, PDUR427	Definitions separated to single requirements

11.3 Changed SWS Items

SWS Item	*Rationale*
PDUR167	Clarification
PDUR171	Clarification
PDUR242	Correction (BooleanParamDef instead of StringParamDef)
PDUR255	FIFO of size 1 shall be considered as a single buffer
PDUR258	FIFO of size 1 shall be considered as a single buffer Clarification for rule (a): TxConfP shall only be set if <Lo>If_Transmit returns with success
PDUR244	FIFO of size 1 shall be considered as a single buffer (parameter depth); Clarification (parameter length)
PDUR100	Development error PDUR_E_IF_TX_BUFFER_MISMATCH and PDUR_E_TP_ BUFFER_SIZE_LIMIT removed, “idle” removed at PDUR_E_TP_TX_REQ_REJECTED
PDUR225	Plausibility check for TpChunkSize added
PDUR248	Correction (SduLength required for gateway operation)
PDUR132	SchM_PduR.h, MemMap.h added to include file structure
PDUR142	Clarification: Transmit confirmation is configurable for unbuffered I-PDUs
PDUR194	Clarification: this function must not be called in the context of CanIf_Transmit.
PDUR196	Clarification: this function must not be called in the context of FrIf_Transmit.
PDUR198	Clarification: this function must not be called in the context of LinIf_Transmit.
PDUR102	Correction: Det_ReportError parameter InstanceId added
PDUR425	Reference to ZERO_COST_OPERATION removed as a result of Bug#18912
PDUR352	Redundant requirement with no additional functionality was Removed.
PDUR374	Redundant requirement with no additional functionality was Removed.
PDUR396	Redundant requirement with no additional functionality was Removed.
PDUR425 PDUR427	Container names changed to VARIANT-PRE-COMPILE VARIANT-POST-BUILD

11.4 Added SWS Items

SWS Item	*Rationale*
PDUR479	New typedefine PduR_CancelReasonType is added.
PDUR480	New typedefine PduR_ParameterValueType is added.
PDUR481	New API PduR_CancelTransmitRequest is added.
PDUR482	New API PduR_ChangeParameterRequest is added.
PDUR483	New Callback PduR_CanTpChangeParameterConfirmation is added.
PDUR484	New Callback PduR_FrTpChangeParameterConfirmation is added.
PDUR485	New Callback PduR_LinTpChangeParameterConfirmation is added.
PDUR486	New requirement is added to Minimum Routing section.
PDUR403	UML model linking of PduR_LinTpTxConfirmation, Result values are updated.
PDUR397	UML model linking of PduR_LinTpRxIndication, Result values are updated
PDUR381	UML model linking of PduR_FrTpTxConfirmation, Result values are updated.
PDUR375	UML model linking of PduR_FrTpRxIndication, Result values are updated.
PDUR359	UML model linking of PduR_CanTpTxConfirmation, Result values are updated.
PDUR353	UML model linking of PduR_CanTpRxIndication, Result values are updated.

12 Changes during SWS Improvements by Technical Office

12.1 Deleted SWS Items

SWS Item	*Rationale*
PDUR169	Requirement on the PDU Router module's environment.
PDUR171	No requirement on the module.
PDUR239	No requirement but information
PDUR251	No requirement on the module.
PDUR165	Requirement on Zero Cost Operation

12.2 Replaced SWS Items

SWS Item	*replaced by SWS Item*	*Rationale*
PDUR102	PDUR331, PDUR332	Made requirement atomic
PDUR108	PDUR335, PDUR336, PDUR337	Made requirement atomic
PDUR134	PDUR295, PDUR295	Made requirement atomic
PDUR142	PDUR301, PDUR302	Made requirement atomic
PDUR143	PDUR432, PDUR433	Made requirement atomic
PDUR158	PDUR292, PDUR293	Made requirement atomic
PDUR167	PDUR428, PDUR429	Made requirement atomic
PDUR170	PDUR436, PDUR437	Made requirement atomic
PDUR172	PDUR314, PDUR315, PDUR316, PDUR317, PDUR318, PDUR438	Made requirement atomic
PDUR174	PDUR324, PDUR325, PDUR326, PDUR327	Made requirement atomic
PDUR175	PDUR297, PDUR298	Made requirement atomic
PDUR178	PDUR319, PDUR320	Made requirement atomic
PDUR181	PDUR439, PDUR440	Made requirement atomic
PDUR182	PDUR454, PDUR455	Made requirement atomic
PDUR183	PDUR469, PDUR470	Made requirement atomic
PDUR184	PDUR441, PDUR442	Made requirement atomic
PDUR185	PDUR456, PDUR457	Made requirement atomic
PDUR186	PDUR471, PDUR472	Made requirement atomic
PDUR187	PDUR443, PDUR444	Made requirement atomic
PDUR188	PDUR458, PDUR459	Made requirement atomic
PDUR189	PDUR473, PDUR474	Made requirement atomic
PDUR190	PDUR445, PDUR446, PDUR447	Made requirement atomic
PDUR191	PDUR460, PDUR461, PDUR462	Made requirement atomic
PDUR192	PDUR475, PDUR476, PDUR477	Made requirement atomic
PDUR195	PDUR448, PDUR449	Made requirement atomic
PDUR196	PDUR450, PDUR451	Made requirement atomic
PDUR197	PDUR463, PDUR464	Made requirement atomic
PDUR198	PDUR465, PDUR466	Made requirement atomic
PDUR199	PDUR452, PDUR453	Made requirement atomic
PDUR200	PDUR467, PDUR468	Made requirement atomic
PDUR202	PDUR409, PDUR410, PDUR411	Made requirement atomic

PDUR203	PDUR328, PDUR329, PDUR330	Made requirement atomic
PDUR208	PDUR434, PDUR435	Made requirement atomic
PDUR209	PDUR430, PDUR431	Made requirement atomic
PDUR210	PDUR299, PDUR300	Made requirement atomic
PDUR211	PDUR303, PDUR304, PDUR305, PDUR306, PDUR307	Made requirement atomic
PDUR254	PDUR310, PDUR311	Made requirement atomic
PDUR257	PDUR312, PDUR313	Made requirement atomic
PDUR260	PDUR308, PDUR309	Made requirement atomic

12.3 Changed SWS Items

Many requirements have been changed to improve understandability without changing the technical contents.

SWS Item	*Rationale*

12.4 Added SWS Items

SWS Item	*Rationale*
PDUR333	UML model linking of imported types
PDUR334	UML model linking of PduR_Init
PDUR338	UML model linking of PduR_GetVersionInfo
PDUR339	Requirement on PduR_GetVersionInfo
PDUR340	Requirement on PduR_GetVersionInfo
PDUR341	UML model linking of PduR_GetConfigurationId
PDUR342	Requirement on PduR_GetConfigurationId
PDUR343	UML model linking of PduR_CanIfRxIndication
PDUR344	Requirement on PduR_CanIfRxIndication
PDUR345	Requirement on PduR_CanIfRxIndication
PDUR346	UML model linking of PduR_CanIfTxConfirmation
PDUR347	Requirement on PduR_CanIfTxConfirmation
PDUR348	Requirement on PduR_CanIfTxConfirmation
PDUR349	Requirement on PduR_CanIfTxConfirmation
PDUR350	UML model linking of PduR_CanTpProvideRxBuffer
PDUR351	Requirement on PduR_CanTpProvideRxBuffer
PDUR352	Requirement on PduR_CanTpProvideRxBuffer
PDUR354	Requirement on PduR_CanTpRxIndication
PDUR355	Requirement on PduR_CanTpRxIndication
PDUR356	UML model linking of PduR_CanTpProvideTxBuffer
PDUR357	Requirement on PduR_CanTpProvideTxBuffer
PDUR358	Requirement on PduR_CanTpProvideTxBuffer
PDUR360	Requirement on PduR_CanTpTxConfirmation
PDUR361	Requirement on PduR_CanTpTxConfirmation
PDUR362	UML model linking of PduR_FrIfRxIndication
PDUR363	Requirement on PduR_FrIfRxIndication
PDUR364	Requirement on PduR_FrIfRxIndication
PDUR365	UML model linking of PduR_FrIfTxConfirmation

PDUR366	Requirement on PduR_FrIfTxConfirmation
PDUR367	Requirement on PduR_FrIfTxConfirmation
PDUR368	Requirement on PduR_FrIfTxConfirmation
PDUR369	UML model linking of PduR_FrIfTriggerTransmit
PDUR370	Requirement on PduR_FrIfTriggerTransmit
PDUR371	Requirement on PduR_FrIfTriggerTransmit
PDUR372	UML model linking of PduR_FrTpProvideRxBuffer
PDUR373	Requirement on PduR_FrTpProvideRxBuffer
PDUR374	Requirement on PduR_FrTpProvideRxBuffer
PDUR376	Requirement on PduR_FrTpRxIndication
PDUR377	Requirement on PduR_FrTpRxIndication
PDUR378	UML model linking of PduR_FrTpProvideTxBuffer
PDUR379	Requirement on PduR_FrTpProvideTxBuffer
PDUR380	Requirement on PduR_FrTpProvideTxBuffer
PDUR382	Requirement on PduR_FrTpTxConfirmation
PDUR383	Requirement on PduR_FrTpTxConfirmation
PDUR384	UML model linking of PduR_LinIfRxIndication
PDUR385	Requirement on PduR_LinIfRxIndication
PDUR386	Requirement on PduR_LinIfRxIndication
PDUR387	UML model linking of PduR_LinIfTxConfirmation
PDUR388	Requirement on PduR_LinIfTxConfirmation
PDUR389	Requirement on PduR_LinIfTxConfirmation
PDUR390	Requirement on PduR_LinIfTxConfirmation
PDUR391	UML model linking of PduR_LinIfTriggerTransmit
PDUR392	Requirement on PduR_LinIfTriggerTransmit
PDUR393	Requirement on PduR_LinIfTriggerTransmit
PDUR394	UML model linking of PduR_LinTpProvideRxBuffer
PDUR395	Requirement on PduR_LinTpProvideRxBuffer
PDUR396	Requirement on PduR_LinTpProvideRxBuffer
PDUR398	Requirement on PduR_LinTpRxIndication
PDUR399	Requirement on PduR_LinTpRxIndication
PDUR400	Requirement on PduR_LinTpProvideTxBuffer
PDUR401	UML model linking of PduR_LinTpProvideTxBuffer
PDUR402	UML model linking of PduR_LinTpProvideTxBuffer
PDUR404	Requirement on PduR_LinTpTxConfirmation
PDUR405	Requirement on PduR_LinTpTxConfirmation
PDUR406	UML model linking of PduR_ComTransmit
PDUR407	Requirement on PduR_ComTransmit
PDUR408	UML model linking of PduR_DcmTransmit
PDUR412	Requirement on PduR_DcmTransmit
PDUR413	UML model linking of PduR_IpdumTransmit
PDUR414	Requirement on PduR_IpdumTransmit
PDUR415	UML model linking of PduR_IpdumTxConfirmation
PDUR416	Requirement on PduR_IpdumTxConfirmation
PDUR417	Requirement on PduR_IpdumTxConfirmation
PDUR418	Requirement on PduR_IpdumTxConfirmation
PDUR419	UML model linking of PduR_IpdumRxIndication
PDUR420	Requirement on PduR_IpdumRxIndication
PDUR421	Requirement on PduR_IpdumRxIndication
PDUR422	Requirement on PduR_IpdumRxIndication
PDUR423	UML model linking of mandatory interfaces
PDUR424	UML model linking of optional interfaces
PDUR425	Every variant gets a requirement ID
PDUR426	Every variant gets a requirement ID
PDUR427	Every variant gets a requirement ID
PDUR478	An Integrator's remark added for the implementation of critical sections