Document Title	Specification of PDU Router
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.0.1
Document Status	Final

Document Change History

Date	Version	Changed by	Change Description
26.06.2006	2.0.1	AUTOSAR Administration	Layout Adaptations
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 11
20.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

This specification as released by the AUTOSAR Development Partnership is intended **for the purpose of information only**. The use of material contained in this specification requires membership within the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership . The AUTOSAR Development Partnership will not be liable for any use of this Specification.

Following the completion of the development of the AUTOSAR Specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

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 AUTOSAR compliance certification 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 provides services for routing of I-PDUs between the following modules:

- communication interface modules (e.g. LIN, CAN, and FlexRay)
- 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. LIN, CAN, or FlexRay) or I-PDU Multiplexer
- I-PDU Multiplexer and communication interface modules (e.g. LIN, CAN, or FlexRay)

PDUs are identified by static PDU IDs. The PDU Router determines the destination of a PDU by using the PDU ID and a static configuration table. I-PDUs (Interaction Layer Protocol Data Units) 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 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 provides an API for modules below the PDU Router (communication interface modules and transport protocol modules) and an API for modules directly above (e.g. DCM and COM) [1]. Furthermore the PDU Router 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 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 is bypassed. Figure 1 gives an overview of the AUTOSAR communication structure.

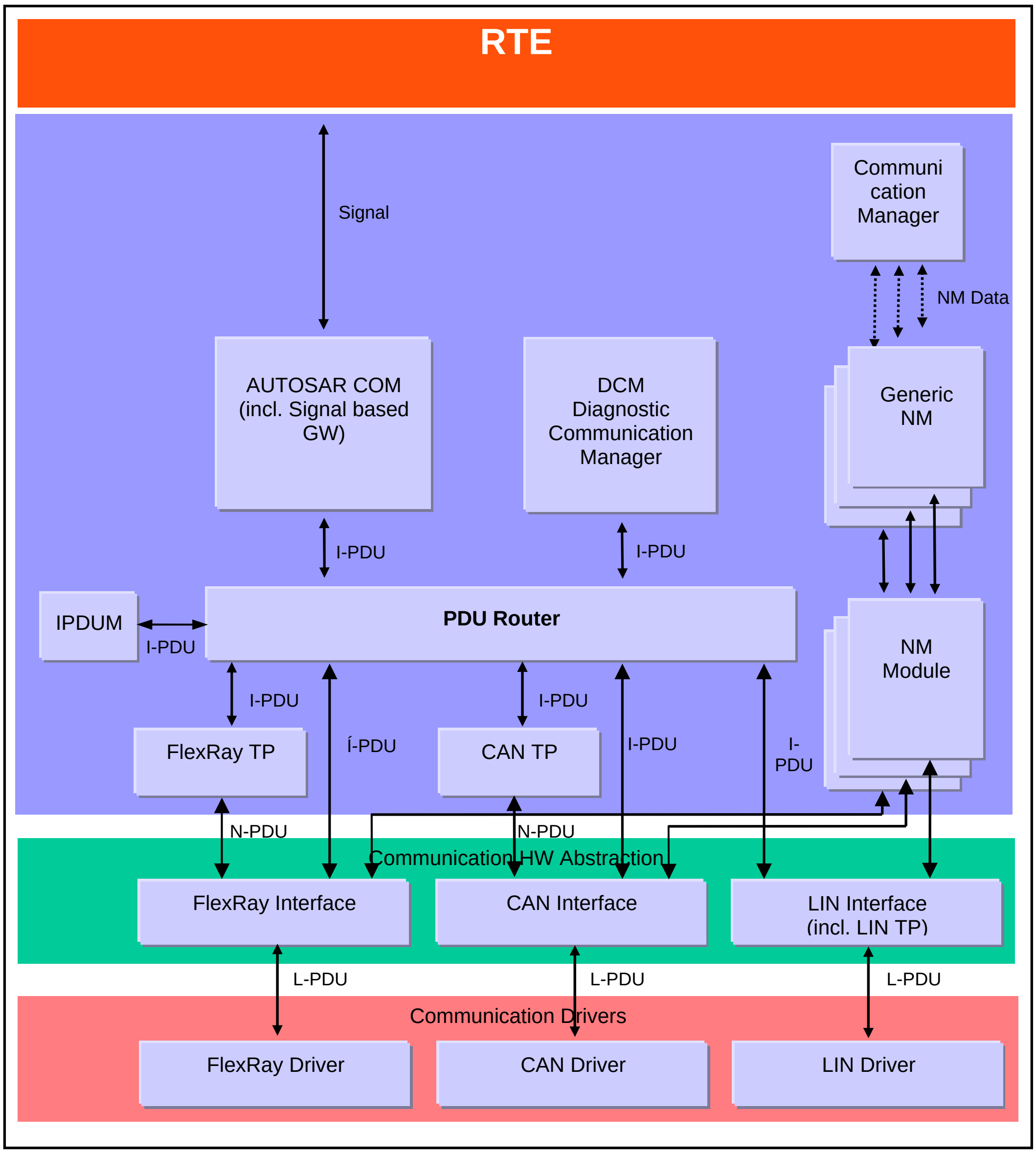

Figure 1: Communication Structure

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

The detailed PDU Router 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 to the related module above the PDU Router (e.g.: COM, DCM, …) or the IPDUM.

- The **PDU Router LO Translation (PRLOT)**: Translation of PDU IDs and API of the PDU Router to the related module below the PDU Router (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 tablesare corrupted. The minimum routing settings are separated from the PDU Router routing tables and cannot be changed after build-time.

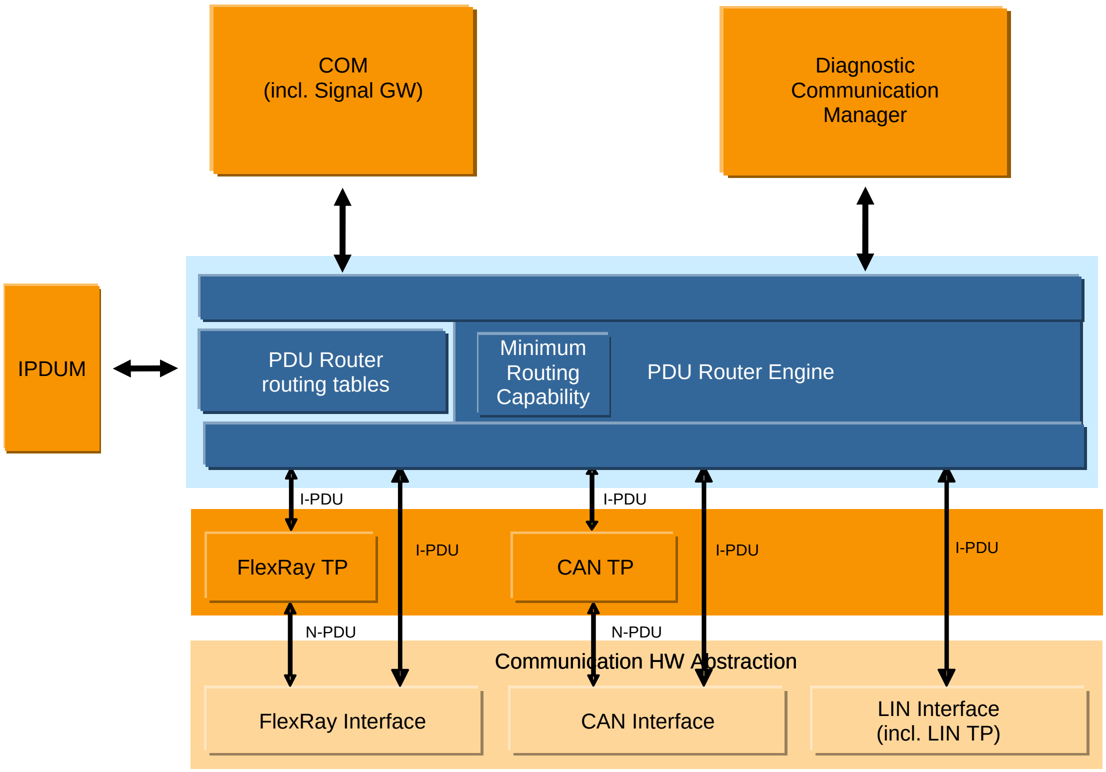

Figure 2: Detailed PDU Router Structure

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 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

3.2 Related standards and norms

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

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

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

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

4 Constraints and assumptions

4.1 Limitations

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

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

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

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

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

6. The PDU Router 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 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 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 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 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

PDUR158: General PDU Router definitions shall be defined in PduR.h and type definitions shall be defined in PduR_Types.h.

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

PDUR216: Due to the high number of communication modules related to the PDU Router, the APIs 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 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` and all `PduR_<module>.h, <module>.h` and `Det.h` if the related pre-compile time configuration parameter is enabled (e.g. PDUR_FRIF_SUPPORT for PduR_FrIf.h).

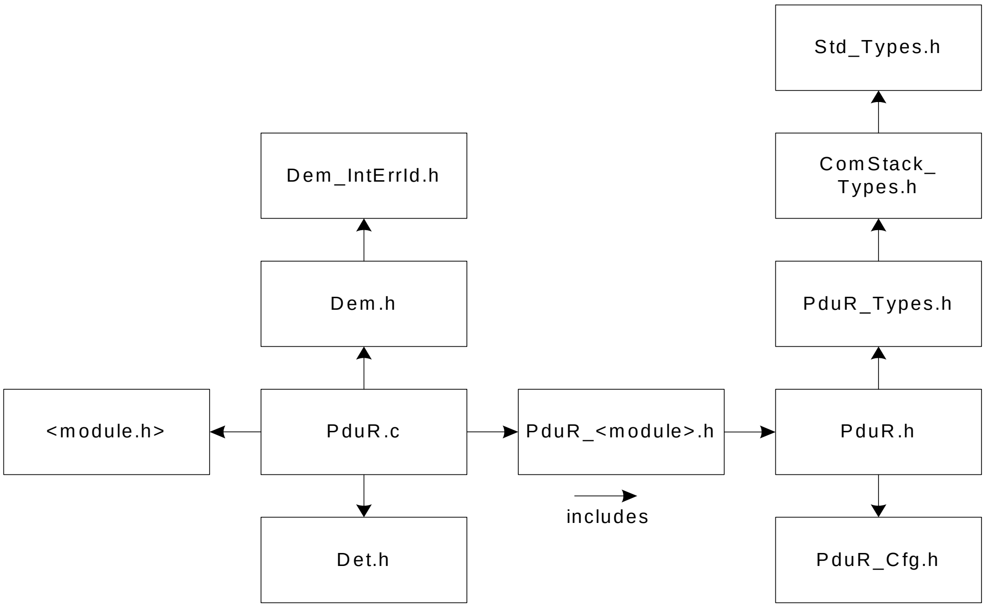

Figure 3: File Structure

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, PDUR102, PDUR103
[BSW00338] Detection and Reporting of development errors	PDUR100, PDUR101, PDUR102, 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	PDUR102, 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, PDUR261, PDUR262, PDUR263, PDUR264, PDUR265, PDUR266, PDUR267, PDUR268, PDUR269, PDUR270, PDUR271, PDUR272, PDUR273, PDUR274, PDUR275, PDUR276, PDUR277, PDUR278, PDUR279, PDUR282, PDUR283, PDUR247, PDUR248, PDUR249
[BSW004] Version check	Implementation requirement
[BSW00400] Selectable Post-build time parameters	not applicable
[BSW00402] Published information	PDUR236

Requirement	*Satisfied by*
[BSW00404] Reference to post build time configuration	PDUR241, PDUR226
[BSW00405] Reference to multiple configuration sets	not applicable
[BSW00406] Check module initialization	PDUR174, 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	PDUR108
[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	PDUR165, 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	PDUR239
[BSW00346] Basic set of module files	PDUR132, PDUR226
[BSW00379] Module identification	PDUR217
[BSW00415] User dependent include files	PDUR216, PDUR158, PDUR132

[BSW158] Separation of configuration from implementation	PDUR226, PDUR159, PDUR132

Document: Requirements on Gateway

Requirement	***Satisfied by***
[BSW06001] Protection of routing table	PDUR134
[BSW06002] Updateable Configuration	PDUR134
[BSW06003] Static Routing Rules	PDUR162, PDUR163, PDUR161
[BSW06004] Routing Chronological Order	PDUR175
[BSW06012] Transparent non-TP PDU routing without rate conversion	PDUR160, PDUR166, PDUR168, PDUR169, PDUR170, PDUR171, PDUR193, PDUR194, PDUR195, PDUR196, PDUR197, PDUR198, PDUR199, PDUR200, PDUR201, PDUR237, PDUR209, PDUR211, PDUR252, PDUR253, PDUR254, PDUR255, PDUR256, PDUR257, PDUR258, PDUR259, PDUR260
[BSW06020] PDU Router scalability	PDUR165, PDUR287, PDUR250
[BSW06026] Transparent TP PDU routing	PDUR166, PDUR167, PDUR168, PDUR169, PDUR170, PDUR171, PDUR172, PDUR181, PDUR184, PDUR187, PDUR190, PDUR182, PDUR185, PDUR188, PDUR191, PDUR183, PDUR186, PDUR189, PDUR192, PDUR202, PDUR210, PDUR142
[BSW06029] Routing of Multicast SF-TP PDUs	PDUR164, PDUR210, PDUR142, PDUR172, PDUR181, PDUR184, PDUR190, PDUR182, PDUR185, PDUR191, PDUR183, PDUR186, PDUR192, PDUR206
[BSW06030] Routing of Multicast non TP PDUs without rate conversion	PDUR164, PDUR209, PDUR211, PDUR218, PDUR238
[BSW06032] PDU transmit buffering in PDU Router	PDUR211, PDUR252, PDUR253, PDUR254, PDUR255, PDUR256, PDUR257, PDUR258, PDUR259, PDUR260, PDUR214, PDUR108
[BSW06048] Minimum Routing Capability	PDUR215, PDUR285, PDUR286, PDUR203
[BSW06049] Consistency of PDU Buffer Content	PDUR214
[BSW06097] Configuration identification	PDUR242, PDUR280, PDUR281
[BSW06103] PDU Router Error Handling at unknown PDU-ID	PDUR100, PDUR102, PDUR221
[BSW06104] PDU Router Error Handling at local reception or transmission	PDUR207, PDUR143, PDUR208
[BSW06105] PDU Router Error Handling in gateway case	PDUR178
[BSW06106] PDU Router Error Handling at FIFO handling	PDUR103, PDUR255, PDUR258
[BSW06114] PDU Router API for COM	PDUR201, PDUR218, PDUR251, PDUR216
[BSW06115] PDU Router API for DCM	PDUR202, PDUR206, PDUR251, PDUR216
[BSW06116] PDU Router API for IPDUM	PDUR237, PDUR238, PDUR251, PDUR216
[BSW06117] PDU Router API for bus interfaces	PDUR193, PDUR194, PDUR195, PDUR196, PDUR199, PDUR197, PDUR198, PDUR200, PDUR251, PDUR216 PDUR181, PDUR184, PDUR187, PDUR190, PDUR182, PDUR185, PDUR188, PDUR191, PDUR183, PDUR186, PDUR189, PDUR192

7 Functional Specification

PDUR251: 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 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 to transmit multiplexed I-PDUs or when it is called by the PDU Router 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 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 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 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 shall transfer an I-PDU without modification to the destination module(s).

PDUR161: Within the PDU Router a PDU shall be uniquely identified by a static PDU ID.

PDUR162: All routes (routing rules) shall be defined in static configuration tables.

PDUR134: The PDU Router shall support the update of the routing configuration (i.e. the PDU Router routing tables) post build-time. The PDU Router routing tables shall only be updated when they are not in use. (Remark: The process how the update 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 shall be identifiable by a unique configuration identifier. (Remark: This ID is not used to select one of multiple post-build configuration sets of the PDU Router, but for unique identification of the current PDU Router post-build configuration, e.g. for Diagnostic 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 destination(s) of a PDU shall be identified by using the PDU ID and the static configuration tables.

PDUR175: Every PDU Router operation shall be triggered by another communication module (which is located either below or above the PDU Router). Hence the behavior of all API services of the PDU Router is synchronous although the overall behavior of an API service might be asynchronous (e.g. a transmission request for CAN: PduR_ComTransmit, Com_TxConfirmation).

PDUR164: The PDU Router 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 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 PDU Reception the PDU Router shall transfer received I-PDUs from lower layer modules to upper layer module(s) according to the provided PDU ID.

PDUR167: The receive operation of the PDU Router shall always be triggered by an indication of a lower layer module (communication interface module, transport protocol module). The indication is either invoked by an interrupt or results from polling a communication driver. In case of the transport protocol module the PDU Router 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 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 that the complete I-PDU has been received and the PDU Router 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.

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

7.1.2 PDU Transmission

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

PDUR169: The transmit operation of the PDU Router shall be triggered by a PDU transmit request from an upper layer module. The PDU Router shall forward the request to the lower layer module(s) according to the PDU ID.

PDUR209: 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). In the second case the PDU Router shall forward the request(s) to the upper layer module by calling <Up>_TriggerTransmit. The mechanism used for each target PDU ID is statically configured.

PDUR210: In case of a request for a transmission via a transport protocol, the requested TP module(s) will ask the PDU Router to provide a transmit buffer. For that purpose the PDU Router shall forward this request to the related upper layer module by invoking <Up>_ProvideTxBuffer. In case of a multicast single frame TP transmission only the first transmit buffer request shall be forwarded to the upper layer module and the returned transmit buffer shall also be provided to the other TP modules.

PDUR142: The transmit operations are always asynchronous. This means that a transmission service request returns immediately after the I-PDU has been passed to the lower layer module. The PDU Router will be notified by the lower layer module 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 I-PDUs which are not sent via TP. 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. In case of a multicast single frame TP transmission only the transmit confirmation of the last TP module shall be forwarded to the upper layer module as the buffer must not be released before.

PDUR143: A transmission request may be rejected by a called lower layer module. This rejection is indicated by the return value of the call. The PDU Router itself shall not perform any error handling and shall simply return the error to the upper layer module. Appropriate error handling is in the responsibility of the upper layer module. In case of a multicast transmission request an error shall be returned if at least one of the related transmission requests returns an error.

PDUR208: If a transmitting TP module reports an error, the PDU Router shall not perform any error handling and shall simply forward the error to the upper layer module via <Up>_TxConfirmation. In case of a multicast single frame TP transmission only the first reported error shall be considered and the error shall be forwarded to the upper layer module in the context of the transmit confirmation of the last TP module.

7.1.3 PDU Gateway

PDUR170: The PDU Router shall support routing of I-PDUs between communication interface modules without rate conversion or between TP modules (PDU Gateway). Therefore the PDU Router has to forward an I-PDU received from one lower layer module (source network) to the lower layer modules (destination networks) identified by the provided PDU ID.

PDUR171: The PDU gateway operation shall be triggered by receiving an appropriate I-PDU indicated by a lower layer module (communication interface module, transport protocol module). The indication is either invoked by an interrupt or results from polling a communication driver.

PDUR211: When receiving an I-PDU from a source interface module which shall be forwarded to at least one destination interface module the PDU Router shall do this by calling <Lo>If_Transmit of the destination interface module(s). The PDU Router shall provide a dedicated PDU transmit buffer for each destination I-PDU, which is configured to use TriggerTransmit data provision (described by PDUR209). The transmit buffer can be statically configured as (a) a single buffer with overwrite behavior or as (b) FIFO of size n with flush-on-overrun behavior (i.e.in case of an buffer overrun, the FIFO shall be flushed and the new I-PDU shall be used as first entry of the FIFO). The PDU Router shall also support a FIFO for I-PDUs which are configured to use Direct data provision (even though this is only required in very special cases).

PDUR260: PDU transmit buffers which are configured as single buffers shall be initialized by the PDU Router initialization function. Thereby the related configured default value shall be copied to the transmit buffer. A PDU Router internal transmit request for an I-PDU which has a single buffer configured as PDU transmit buffer shall be processed in the following way: the new I-PDU shall be copied to the transmit buffer and <Lo>If_Transmit of the related interface module shall be called. The I-PDU stored in the transmit buffer shall be used by the related PduR_<Lo>IfTriggerTransmit call.

PDUR252: A transmit confirmation shall be configured for each I-PDU which has a FIFO configured as PDU transmit buffer (even if it belongs to a multicast transmission).

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

PDUR254: At least two values shall be maintained 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. The FIFOs shall be initialized by the PDU Router initialization function. Thereby TxConfP shall be cleared; the related configured default value shall be copied to the FIFO and TxIdx shall be set to this entry.

PDUR255: A PDU Router internal transmit request for an I-PDU which has a TT-FIFO configured as PDU transmit buffer shall be processed according to the following rules:

(a) If TxConfP is not set, the new I-PDU shall replace the FIFO entry specified by TxIdx, <Lo>If_Transmit of the related interface module shall be called and if it returns with success (i.e. E_OK), TxConfP shall be set.
(b) If TxConfP is set and the FIFO is not full, the new I-PDU shall be added to the FIFO.
(c) If TxConfP is set and the FIFO is full, the FIFO shall be flushed (i.e. all entries shall be removed from the FIFO, TxIdx shall be initialized and TxConfP shall be cleared), the error `PDUR_E_PDU_INSTANCE_LOST` shall be reported to DEM if the FIFO is of size 2 or more, and the new I-PDU shall be processed according to rule (a).

PDUR256: A transmit confirmation for an I-PDU which has a TT-FIFO configured as PDU transmit buffer shall be processed according to the following rules:
(a) If TxConfP is not set, the confirmation shall be ignored.
(b) If TxConfP is set and the FIFO contains only one entry, TxConfP shall be cleared.
(c) If TxConfP is set and the FIFO contains more than one entry, the FIFO entry specified by TxIdx shall be removed, TxIdx shall be set to the next FIFO entry and <Lo>If_Transmit of the related interface module shall be called. If it returns without success (i.e. any value other than E_OK), the transmit confirmation has to be processed again according to rule (b) or (c).

PDUR257: 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 shall be maintained. The FIFOs shall be initialized by the PDU Router initialization function. Thereby TxConfP shall be cleared.

PDUR258: A PDU Router internal transmit request for an I-PDU which has a D-FIFO configured as PDU transmit buffer shall be processed according to the following rules:
(a) If TxConfP is not set, <Lo>If_Transmit of the related interface module shall be called with the new I-PDU and the TxConfP shall be set.
(b) If TxConfP is set and the FIFO is not full, the new I-PDU shall be added to the FIFO.
(c) If TxConfP is set and the FIFO is full, the FIFO shall be flushed (i.e. all entries shall be removed from the FIFO and TxConfP shall be cleared), the error `PDUR_E_PDU_INSTANCE_LOST` shall be reported to DEM if the FIFO is of size 2 or more, and the new I-PDU shall be processed according to rule (a).

PDUR259: A transmit confirmation for an I-PDU which has a D-FIFO configured as PDU transmit buffer shall be processed according to the following rules:
(a) If TxConfP is not set, the confirmation shall be ignored.
(b) If TxConfP is set and the FIFO is empty, TxConfP shall be cleared.
(c) If TxConfP is set and the FIFO is not empty, <Lo>If_Transmit of the related interface module shall be called with the next FIFO entry. Thereafter this entry shall be removed from the FIFO. If <Lo>If_Transmit returns without success (i.e. any value other than E_OK), the transmit confirmation has to be processed again according to rule (b) or (c).

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

PDUR172: In case of routing between TP modules forwarding of the I-PDU shall be started before the full I-PDU is received (“routing on-the-fly”). For that purpose the PDU Router shall provide a small receive buffer when requested via PduR_<Lo>TpProvideRxBuffer. The buffer size 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 function PduR_<Lo>TpProvideRxBuffer will be called more than once. 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 single, 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 shall call <Lo>Tp_Transmit when the first receive buffer is released by the receiving TP module (either within PduR_<Lo>TpProvideRxBuffer or PduR_<Lo>TpRxIndication). The PDU Router shall release the related transmit buffer within PduR_<Lo>TpProvideTxBuffer or PduR_<Lo>TpTxConfirmation respectively. In case of a multicast single frame TP gateway the PDU Router 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.

PDUR178: The PDU Router 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. Whenever a TP module which is part of an active TP gateway operation reports an error, the PDU Router 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

PDUR165: The PDU Router shall support a zero cost operation mode by using macros instead of functions and scale down to no size in case all of the following six conditions are true (1) there is only one interface module and (2) no PDU gateway functionality is needed and (3) no multicast PDU is configured and (4) there is at most one upper layer module which communicates via the interface module and (5) no IPDUM is used, and (6) there is at most one upper layer module which communicates via a TP module. If all of these conditions are fulfilled, every routing path is implicitly defined and the pre-compile time configuration parameter PDUR_ZERO_COST_OPERATION may be enabled.

PDUR287: If the pre-compile time configuration parameter PDUR_ZERO_COST_OPERATION 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 PDUR_ZERO_COST_OPERATION. In the latter case the configuration parameters PDUR_SINGLE_IF and PDUR_SINGLE_TP 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 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) PDUR_MINIMUM_ROUTING_UP_MODULE, and PDUR_MINIMUM_ROUTING_LO_MODULE to specify the upper and lower layer modules to be used for minimum routing,
(b) PDUR_MINIMUM_ROUTING_LO_RXPDUID and PDUR_MINIMUM_ROUTING_UP_RXPDUID to specify the RxPduIds for PDU Reception and
(c) PDUR_MINIMUM_ROUTING_UP_TXPDUID and PDUR_MINIMUM_ROUTING_LO_TXPDUID to specify the TxPduIds for PDU Transmission.

PDUR286: Minimum routing shall always have precedence over routing according to the post-build time configurable PDU Router routing tables. In case of zero cost operation according to PDUR165, every routing path is implicitely defined and no routing decisions shall be performed by the PDU Router at runtime.

Note: Minimum routing will be performed in the online state (PDUR_ONLINE) as well as in the reduced state (PDUR_REDUCED), see PDUR203.

7.4 Reentrance

PDUR239: The reentrance 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

PDUR174: As shown in Figure 4 the PDU Router shall consist of three states, `PDUR_UNINIT`, `PDUR_REDUCED` and `PDUR_ONLINE`. After power up the PDU Router shall be in the `PDUR_UNINIT` state. The PDU Router shall change to the state `PDUR_ONLINE` when the PDU Router has successfully been initialized via PduR_Init(). In case the initialization did not succeed the PDU Router shall change to the state `PDUR_REDUCED`.

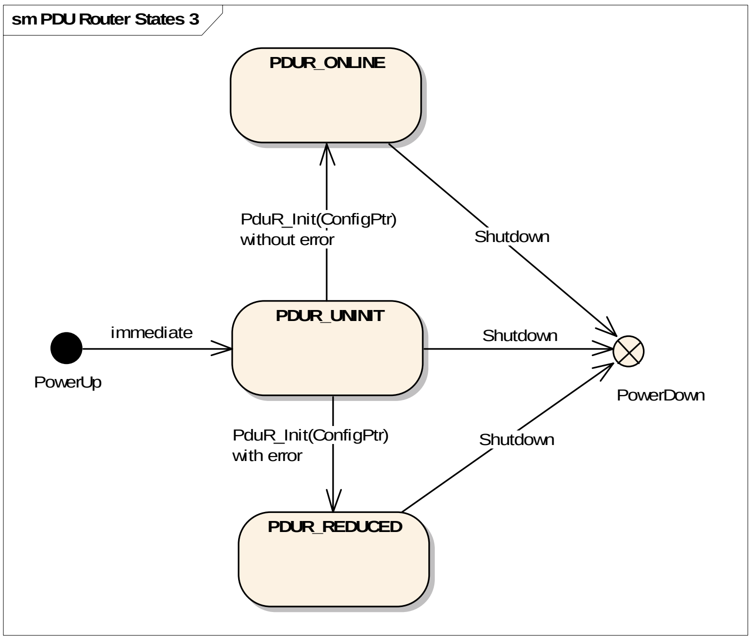

Figure 4: PDU Router states

PDUR203: Routing of PDUs according to the PDU Router routing tables shall only be performed when the PDU Router is in the online state (`PDUR_ONLINE`). Routing of PDUs according to the minimum routing capabilities shall be performed in the states `PDUR_ONLINE` and `PDUR_REDUCED`. No routing shall be performed 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 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 and idle (currently not used in a TP session) PDU identifier	Development	`PDUR_E_TP_TX_REQ_REJECTED`	`0x03`
Transmit buffer size mismatch	Development	`PDUR_E_IF_TX_BUFFER_MISMATCH`	`0x04`
Data pointer (CanSduPtr, FrSduPtr, LinSduPtr or PduInfoPtr) is NULL	Development	`PDUR_E_DATA_PTR_INVALID`	`0x05`
Length of requested TP buffer is larger than the maximum length of all configured TP buffer (TP request impossible)	Development	`PDUR_E_TP_BUFFER_SIZE_LIMIT`	`0x06`
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 PDUR_DEV_ERROR_DETECT (see chapter 10) shall activate or deactivate the detection of all development errors.

PDUR227: If the PDUR_DEV_ERROR_DETECT switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.6 and chapter 8.6.

PDUR233: The detection of production code errors cannot be switched off.

PDUR119: If the PDU Router has not been initialized (PDUR_UNINIT state) all services except PduR_Init() shall report the error PDUR_E_INVALID_REQUEST via the Development Error Tracer (DET) when called.

7.8 Error notification

PDUR102: Detected development errors shall be reported to the Development Error Tracer (DET) if the preprocessor switch PDUR_DEV_ERROR_DETECT is set (see chapter 10). The DET is used to support BSW development and integration but will not be contained in the production code. DET specification [6] defines an API for reporting of development errors but does not specify its implementation. It is up to the software developer and software integrator to choose an optimal strategy for the specific application and testing environment (e.g. set debugger breakpoint, count reported errors, log calls and passed parameters in RAM buffer, send error information via a serial interface to an external logger). When detecting a development error the PDU Router shall report the error to DET by using the DET function shown below and shall thereafter exit the concerned PDU Router function and return an error if possible (e.g. by returning PDUR_E_NOT_OK in case Std_ReturnType is used).

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

ModuleId	Module ID of the PDU Router: 51 decimal (see PDUR217)
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 implementation specification. The classification and enumeration shall be compatible to the errors listed above [PDUR100].

8 API specification

The following paragraphs specify the API of the PDU Router.

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

8.1 Imported types

8.1.1 Standard types

In this chapter all types included from the following files are listed (see [4]):

- Std_Types.h
- ComStack_Types.h

- Std_ReturnType
- Std_VersionInfoType
- PduIdType
- PduLengthType
- PduInfoType
- BufReq_ReturnType
- NotifResultType

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

Type:	Enum	
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:	**PDUR284**: PDU Router states.	

8.2.2 PduR_LConfigType

Type:	Struct	
Range:	--	Implementation dependent structure.
Description:	**PDUR240**: Type of the external data structure containing link-time configuration data of the PDU Router 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

Type:	Struct	
Range:	--	Implementation dependent structure.
Description:	**PDUR241**: Type of the external data structure containing post-build-time configuration data of the PDU Router 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.	

8.3 Function definitions

8.3.1 General functions provided by the PDU Router

8.3.1.1 PduR_Init

Service name:	PduR_Init	

Syntax:	`void PduR_Init`
	`(`
	`    const PduR_PBConfigType * ConfigPtr`
	`)`	

Service ID [hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	non re-entrant	
Parameters (in):	`ConfigPtr`	Pointer to post build configuration
Parameters (out):	None	--
Return value:	None	--
Description:	**PDUR108**: Service for PDU Router initialization. If the configuration parameter PDUR_ZERO_COST_OPERATION is enabled this service shall be realized as an empty function-like macro. Otherwise the Initialization function shall initialize the PDU Router module (e.g. PDU transmit buffers shall be initialized according to PDUR260, PDUR254 or PDUR257 depending on the PDU transmit buffer type). In case this function is called in any state other than `PDUR_UNINIT`, the request shall be ignored and the error `PDUR_E_INVALID_REQUEST` shall be reported to DET if development error detection is enabled. **PDUR106**: After having finished the module initialization without errors, the PDU Router state shall change to `PDUR_ONLINE` state, in case of errors the PDU Router shall change to `PDUR_REDUCED` state and the error `PDUR_E_INIT-_FAILED` shall be reported to DEM.	
Caveats:	None	
Configuration:	--	

8.3.1.2 PduR_GetVersionInfo

Service name:	PduR_GetVersionInfo	

Syntax:	`void PduR_GetVersionInfo`
	`(`
	`      Std_VersionInfoType *versioninfo`
	`)`	

Service ID [hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	none	--
Parameters (out):	`versioninfo`	Pointer to where to store the version information of this module.
Return value:	none	--
Description:	**PDUR234**: This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers. If the configuration parameter PDUR_ZERO_COST_OPERATION is enabled this service shall be realized as a function-like macro. Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	
Caveats:	--	
Configuration:	**PDUR235**: This function shall be pre compile time configurable `On/Off` by the configuration parameter: `PDUR_VERSION_INFO_API`	

8.3.1.3 PduR_GetConfigurationId

Service name:	PduR_GetConfigurationId	

Syntax:	`uint32 PduR_GetConfigurationId`
	`(`
	`      void`
	`)`	

Service ID [hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	none	--
Parameters (out):	none	--
Return value:	none	--
Description:	**PDUR280**: This service returns the unique identifier of the post-build time configuration of the PDU Router (see PDUR242, PDUR_CONFIGURATION_ID). If the configuration parameter PDUR_ZERO_COST_OPERATION is enabled this service shall be realized as a function-like macro which always returns 0.	
Caveats:	--	
Configuration:	--	

8.3.2 Function definitions for CAN interaction

8.3.2.1 PduR_CanIfRxIndication

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 (out):	None	--
Return value:	None	--
Description:	This function is called by the CAN Interface after a CAN L-PDU has been received. **PDUR193**: The PDU Router 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 PDU Router shall process the target PDU according to PDUR258.	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_CANIF_SUPPORT is enabled.	

8.3.2.2 PduR_CanIfTxConfirmation

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 (out):	None	--
Return value:	None	--
Description:	This function is called by the CAN Interface after the PDU has been transmitted on the CAN network. **PDUR194**: The PDU Router 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 PDU Router shall process the confirmation according to PDUR259.
Caveats:	This function might be called in interrupt context (e.g. from CAN transmit interrupt).
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_CANIF_SUPPORT is enabled.

8.3.2.3 PduR_CanTpProvideRxBuffer

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 (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_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:	This service 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. **PDUR181**: The PDU Router shall translate the CanTpRxPduId into the configured target PDU ID and route this request to the configured target function. If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the PDU Router 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 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 this service the receiver (e.g. DCM) is also informed implicitly about a first	

	frame reception or a single frame reception.
Caveats:	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 transport protocol filling the provided buffer will also set the length information contained in *`PduInfoPtr` to the number of bytes that are valid in this buffer. It is expected 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`. This function might be called in interrupt context.
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_CANTP_SUPPORT is enabled.

8.3.2.4 PduR_CanTpRxIndication

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` in case TP reception completed successfully • `NTFRSLT_E_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; see [4] for more details); used to enable unlocking of the receive buffer
Parameters (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR184**: The PDU Router shall translate the CanTpRxPduId into the configured target PDU ID and route this indication to the configured target function. If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the PDU Router 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.
Caveats:	This function might be called in interrupt context.
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_CANTP_SUPPORT is enabled.

8.3.2.5 PduR_CanTpProvideTxBuffer

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 (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_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:	This function is called by the CAN TP for requesting a transmit buffer. The length of the buffer 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`). **PDUR187**: Within this function, the PDU Router shall translate the `CanTpTxPduId` into the configured target PDU ID and route this request to the configured target function. If `CanTpTxPduId` belongs to a gateway operation the	

	PDU Router itself has to provide the requested buffer. Therefor the PDU Router shall use the receive buffer which has previously been filled by the receiving TP module.
Caveats:	This function might be called in interrupt context. In case this service 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.
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_CANTP_SUPPORT is enabled.

8.3.2.6 PduR_CanTpTxConfirmation

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):	`CanTxTpPduId`	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` in case TP transmission completed successfully, • `NTFRSLT_E_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; see [4] for more details); used to enable unlocking of the transmit buffer.
Parameters (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR190**: The PDU Router shall translate the CanTpTxPduId into the configured target PDU ID and route this indication to the configured target function. If CanTpTxPduId belongs to a gateway operation the PDU Router shall use this indication to unlock the transmit buffer. In case of a multicast single frame TP transmission initiated by an upper layer module only the transmit confirmation of the last TP module shall be forwarded to the upper layer module as the buffer must not be released before.	
Caveats:	This function might be called in interrupt context (e.g. from CAN transmit interrupt).	
Configuration:	This function shall only be provided if the pre-compile time configuration	

	parameter PDUR_CANTP_SUPPORT is enabled.

8.3.3 Function definitions for FlexRay interaction

8.3.3.1 PduR_FrIfRxIndication

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 (out):	None	--
Return value:	None	--
Description:	This function is called by the FlexRay Interface after a FlexRay L-PDU has been received. **PDUR195**: The PDU Router shall translate the FrRxPduId 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 PDU transmit buffer, the PDU Router shall process the target PDU according to PDUR260, PDUR255 or PDUR258 depending on the PDU transmit buffer type.	
Caveats:	This function might be called in interrupt context (e.g. from the FlexRay receive interrupt). However, the FlexRay specification does not mandate the existence of a receive interrupt.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRIF_SUPPORT is enabled.	

8.3.3.2 PduR_FrIfTxConfirmation

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.	
5Parameters (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 (out):	None	--
Return value:	None	--
Description:	This function is called by the FlexRay Interface after the PDU has been transmitted on the FlexRay network. **PDUR196**: Within this function, the PDU Router shall translate the FrTxPduId 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 PDU Router shall process the confirmation according to PDUR256 or PDUR259 depending on the FIFO buffer type.	
Caveats:	This function might be called 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.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRIF_SUPPORT is enabled.	

8.3.3.3 PduR_FrIfTriggerTransmit

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 (out):	None	--
Return value:	None	--
Description:	This function is called by the FlexRay Interface for sending out a FlexRay frame. The trigger transmit is initiated by the FlexRay schedule. Whether this function is called or not is statically configured for each PDU. This triggered transmission is mainly used for the static part of FlexRay. **PDUR199**: The PDU Router shall translate the FrTxPduId into the configured target PDU ID and route this trigger to the configured target function (e.g. AUTOSAR COM). If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a PDU transmit buffer, the PDU Router shall copy the data from the PDU transmit buffer to the place specified by FrSduPtr.	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRIF_SUPPORT is enabled.	

8.3.3.4 PduR_FrTpProvideRxBuffer

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 (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_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:	This service 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. **PDUR182**: The PDU Router shall translate the FrTpRxPduId into the configured target PDU ID and route this request to the configured target function. If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the PDU Router 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 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 this service the receiver (e.g. DCM) is also informed implicitly about a first frame reception or a single frame reception.	
Caveats:	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 transport protocol filling the provided buffer will also set the length	

	information contained in `*PduInfoPtr` to the number of bytes that are valid in this buffer. It is expected 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` This function might be called in interrupt context.
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRTP_SUPPORT is enabled.

8.3.3.5 PduR_FrTpRxIndication

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` in case TP reception completed successfully, • `NTFRSLT_E_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; see [4] for more details); used to enable unlocking of the receive buffer
Parameters (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR185**: The PDU Router shall translate the FrTpRxPduId into the configured target PDU ID and route this indication to the configured target function. If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the PDU Router 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.	
Caveats:	This function might be called in interrupt context.	

Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRTP_SUPPORT is enabled.

8.3.3.6 PduR_FrTpProvideTxBuffer

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 (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_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:	This function is called by the FlexRay TP for requesting a transmit buffer. The length of the buffer 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`). **PDUR188**: Within this function, the PDU Router shall translate the `FrTpTxPduId` into the configured target PDU ID and route this request to the configured target function. If `FrTpTxPduId` belongs to a gateway operation the PDU Router itself has to provide the requested buffer. Therefor the PDU Router shall use the reveive buffer which has previously been filled by the receiving TP module.	
Caveats:	This function might be called in interrupt context. In case this service 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.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRTP_SUPPORT is enabled.	

8.3.3.7 PduR_FrTpTxConfirmation

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):	`FrTxTpPduId`	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` in case TP transmission completed successfully, • `NTFRSLT_E_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; see [4] for more details); used to enable unlocking of the transmit buffer.
Parameters (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR191**: The PDU Router shall translate the FrTpRxPduId into the configured target PDU ID and route this indication to the configured target function. If FrTpRxPduId belongs to a gateway operation the PDU Router shall use this indication to unlock the transmit buffer. In case of a multicast single frame TP transmission initiated by an upper layer module only the transmit confirmation of the last TP module shall be forwarded to the upper layer module as the buffer must not be released before.	
Caveats:	This function might be called 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 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.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_FRTP_SUPPORT is enabled.	

8.3.4 Function definitions for LIN interaction

8.3.4.1 PduR_LinIfRxIndication

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 (out):	None	--
Return value:	None	--
Description:	This function is called by the LIN Interface after a LIN L-PDU has been received. **PDUR197**: The PDU Router shall translate the LinRxPduId 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 PDU transmit buffer, the PDU Router shall process the target PDU according to PDUR260, PDUR255 or PDUR258 depending on the PDU transmit buffer type.	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINIF_SUPPORT is enabled.	

8.3.4.2 PduR_LinIfTxConfirmation

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.	
5Parameters (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 (out):	None	--
Return value:	None	--
Description:	This function is called by the LIN Interface after the PDU has been transmitted on the LIN bus. **PDUR198**: The PDU Router shall translate the LinTxPduId 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 PDU Router shall process the confirmation according to PDUR256 or PDUR259 depending on the FIFO buffer type.
Caveats:	This function might be called in interrupt context.
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINIF_SUPPORT is enabled.

8.3.4.3 PduR_LinIfTriggerTransmit

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 (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR200**: The PDU Router shall translate the LinTxPduId into the configured target PDU ID and route this trigger to the configured target function (e.g. AUTOSAR COM).If the target PDU ID belongs to an interface module (gateway case) and is statically configured to use a PDU transmit buffer, the PDU Router shall copy the data from the PDU transmit buffer to the place specified by LinSduPtr.	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINIF_SUPPORT is enabled.	

8.3.4.4 PduR_LinTpProvideRxBuffer

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 (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_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:	This service 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. **PDUR183**: The PDU Router shall translate the LinTpRxPduId into the configured target PDU ID and route this request to the configured target function. If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the PDU Router 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 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 this service the receiver (e.g. DCM) is also informed implicitly about a first frame reception or a single frame reception.	
Caveats:	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 transport protocol filling the provided buffer will also set the length information contained in *PduInfoPtr to the number of bytes that are valid in this buffer. It is expected 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. This function might be called in interrupt context.	

Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINTP_SUPPORT is enabled.

8.3.4.5 PduR_LinTpRxIndication

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` in case TP reception completed successfully; • `NTFRSLT_E_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; see [4] for more details); used to enable unlocking of the receive buffer
Parameters (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR186**: The PDU Router shall translate the LinTpRxPduId into the configured target PDU ID and route this indication to the configured target function. If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case) the PDU Router 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.	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINTP_SUPPORT is enabled.	

8.3.4.6 PduR_LinTpProvideTxBuffer

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 (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_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:	This function is called by the LIN TP for requesting a transmit buffer. The length of the buffer 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`). **PDUR189**: Within this function, the PDU Router shall translate the `LinTpTxPduId` into the configured target PDU ID and route this request to the configured target function. If `LinTpTxPduId` belongs to a gateway operation the PDU Router itself has to provide the requested buffer. Therefor the PDU Router shall use the receive buffer which has previously been filled by the receiving TP module.	
Caveats:	This function might be called in interrupt context. In case this service 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.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINTP_SUPPORT is enabled.	

8.3.4.7 PduR_LinTpTxConfirmation

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):	`LinTxTpPduId`	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` in case TP transmission completed successfully, • `NTFRSLT_E_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; see [4] for more details); used to enable unlocking of the transmit buffer.
Parameters (out):	None	--
Return value:	None	--
Description:	This function 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. **PDUR192**: The PDU Router shall translate the LinTpRxPduId into the configured target PDU ID and route this indication to the configured target function. If LinTpRxPduId belongs to a gateway operation the PDU Router shall use this indication to unlock the transmit buffer. In case of a multicast single frame TP transmission initiated by an upper layer module only the transmit confirmation of the last TP module shall be forwarded to the upper layer module as the buffer must not be released before.	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_LINTP_SUPPORT is enabled.	

8.3.5 Function definitions for COM interaction

8.3.5.1 PduR_ComTransmit

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 (out):	`None`	--
Return value:	`E_OK`	Transmit request has been accepted
	`E_NOT_OK`	Transmit request has not been accepted
Description:	This function is called by AUTOSAR COM to request a transmission. **PDUR201**: The PDU Router 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 PDU Router shall return E_NOT_OK.	
Caveats:	None	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_COM_SUPPORT is enabled.	

8.3.6 Function definitions for DCM interaction

8.3.6.1 PduR_DcmTransmit

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 (out):	`None`	--
Return value:	`E_OK`	Transmit request has been accepted
	`E_NOT_OK`	Transmit request has not been accepted
Description:	This function is called by the DCM to request a transmission.	

	PDUR202: The PDU Router shall translate the DcmTxPduId into the configured target PDU ID and route this transmit request to the configured target TP module. Within the parameter PduInfoPtr, only the SduLength information shall be used. The pointer to the data is undefined and must not be used. For a TP transmission request this service call 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 DcmTxPduId 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 PDU Router shall return E_NOT_OK.
Caveats:	None
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_DCM_SUPPORT is enabled.

8.3.7 Function definitions for IPDUM interaction

8.3.7.1 PduR_IpdumTransmit

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)	`IpdumUpTxPduId`	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 (out):	`None`	--
Return value:	`E_OK`	Transmit request has been accepted
	`E_NOT_OK`	Transmit request has not been accepted
Description:	This function 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 PDU Router shall translate the IpdumUpTxPduId into the configured target PDU ID and route this transmit request to the configured target interface module. **PDUR238**: If IpdumUpTxPduId represents a group of PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error the PDU Router shall return E_NOT_OK.	
Caveats:	None	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_IPDUM_SUPPORT is enabled.	

8.3.7.2 PduR_IpdumTxConfirmation

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 (out):	`None`	--
Return value:	`None`	--
Description:	This function is called by IPDUM (acting as a lower layer module) after the PDU has been transmitted. The PDU Router shall translate the IpdumLoTxPduId into the configured target PDU ID and route this confirmation to the configured upper layer module (e.g. COM).	
Caveats:	This function might be called in interrupt context	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_IPDUM_SUPPORT is enabled.	

8.3.7.3 PduR_IpdumRxIndication

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 (out):	None	--
Return value:	None	--
Description:	This function is called by the IPDUM (acting as a lower layer module) after the PDU has been received. The PDU Router shall translate the IpdumLoRxPduId into the configured target PDU ID and route this indication to the configured upper layer module (e.g. COM).	
Caveats:	This function might be called in interrupt context.	
Configuration:	This function shall only be provided if the pre-compile time configuration parameter PDUR_IPDUM_SUPPORT is enabled.	

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 mandatory interfaces required from other modules.

API function	***Module***	***Description***
Dem_ReportError	Dem	Report a production mode error

8.5.2 Optional Interfaces

This chapter defines all interfaces which are only required from other modules if the related pre-compile time configuration parameter is enabled.

API function	***Module***	***Description***	***Configuration parameter (description see chapter 10)***
CanIf_Transmit	CanIf	Requests a transmission on the Can bus	PDUR_CANIF_SUPPORT
FrIf_Transmit	FrIf	Requests a transmission on the FlexRay bus	PDUR_FRIF_SUPPORT
LinIf_Transmit	LinIf	Requests a transmission on the LIN bus	PDUR_LINIF_SUPPORT
CanTp_Transmit	CanTp	Requests a TP transmission on the Can bus	PDUR_CANTP_SUPPORT
FrTp_Transmit	FrTp	Requests a TP transmission on the FlexRay bus	PDUR_FRTP_SUPPORT
LinTp_Transmit	LinIf	Requests a TP transmission on the LIN bus	PDUR_LINTP_SUPPORT
Com_RxIndication	Com	Take received I-PDU	PDUR_COM_SUPPORT
Com_TriggerTransmit	Com	Copy I-PDU to transmit buffer specified by SduPtr	PDUR_COM_SUPPORT
Com_TxConfirmation	Com	Transmit confirmation	PDUR_COM_SUPPORT
Dcm_ProvideRxBuffer	Dcm	Provide reception buffer for TP message	PDUR_DCM_SUPPORT
Dcm_RxIndication	Dcm	Indicates end of reception	PDUR_DCM_SUPPORT
Dcm_ProvideTxBuffer	Dcm	Provide transmit buffer for TP message	PDUR_DCM_SUPPORT

Dcm_TxConfirmation	Dcm	Transmit confirmation	PDUR_DCM_SUPPORT
Ipdum_Transmit	Ipdum	Request a transmission of a multiplexed PDU	PDUR_IPDUM_SUPPORT
Ipdum_TxConfirmation	Ipdum	Transmit confirmation	PDUR_IPDUM_SUPPORT
Ipdum_TriggerTransmit	Ipdum	Copy I-PDU to transmit buffer specified by SduPtr	PDUR_IPDUM_SUPPORT
Ipdum_RxIndication	Ipdum	Take received multiplexed I-PDU	PDUR_IPDUM_SUPPORT
Det_ReportError	Det	Development error notification	PDUR_DEV_ERROR_DETECT

8.5.3 Configurable interfaces

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

8.6 API parameter checking

PDUR221: The PDU identifier shall be within the specified range and shall be configured to be used by the PDU Router either for minimum routing (`PDUR_ONLINE` and `PDUR_REDUCED` state) or for routing according to the post-build routing tables (`PDUR_ONLINE` state). Otherwise `PDUR_E_PDU_ID_INVALID` shall be reported to DET.

PDUR222: `ConfigPtr` of initialization function `PduR_Init()` shall not be NULL. Otherwise `PDUR_E_CONFIG_PTR_INVALID` shall be reported to DET.

PDUR223: A data pointer (`CanSduPtr`, `FrSduPtr`, `LinSduPtr` or `PduInfoPtr`) shall not be NULL. Otherwise `PDUR_E_DATA_PTR_INVALID` shall be reported to DET.

PDUR224: The requested TP buffer size of gateway operation shall not be larger than the maximum length of all configured TP buffer. Otherwise `PDUR_E_TP_BUFFER_SIZE_LIMIT` shall be reported to the DET.

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).

9.1 Initialization

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

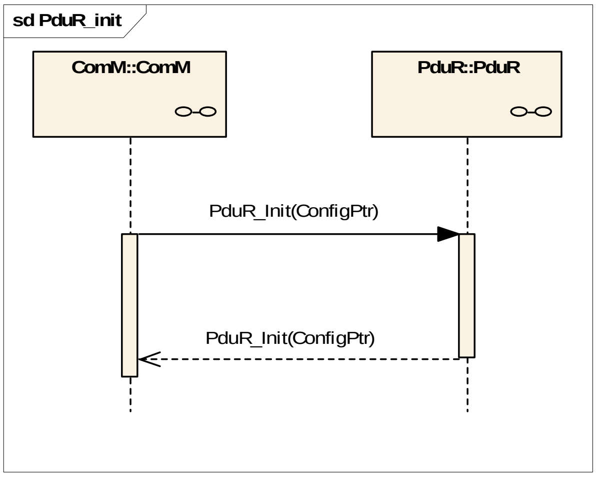

Figure 5: Initialization

9.2 PDU Reception

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

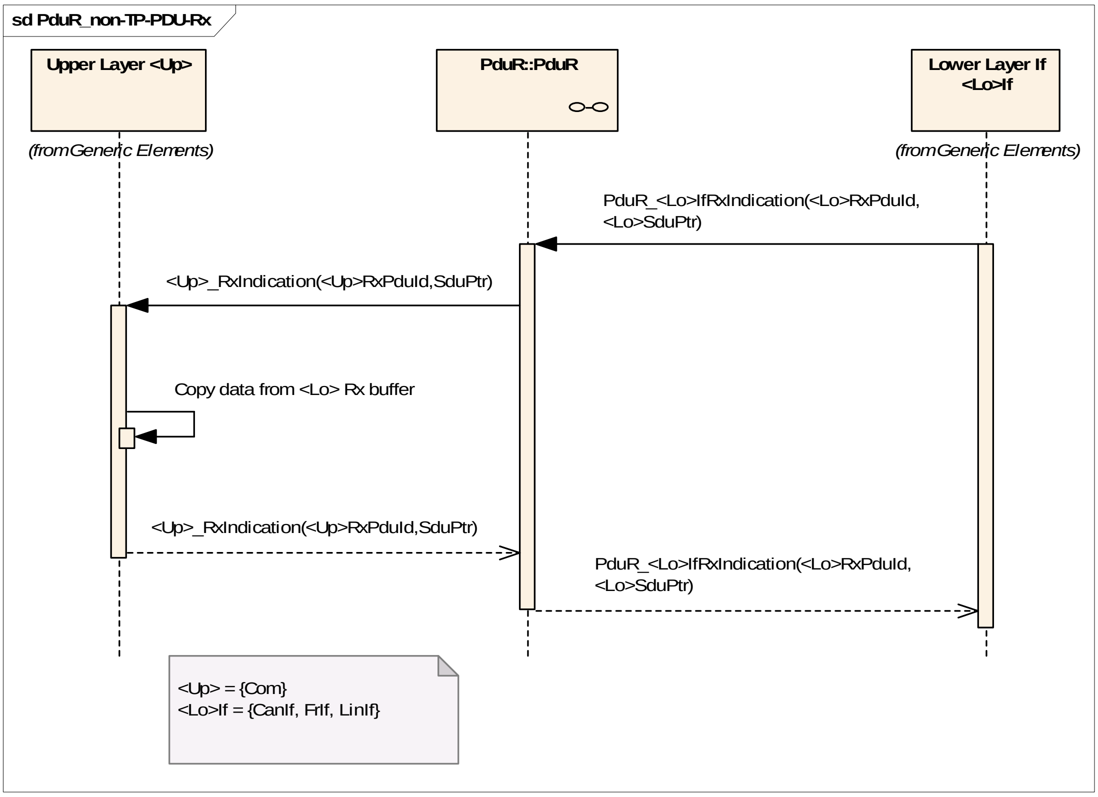
sd PduR_non-TP-PDU-Rx
Upper Layer <Up>
(fromGeneric Elements)
PduR::PduR
Lower Layer If <Lo>If
(fromGeneric Elements)
PduR_<Lo>IfRxIndication(<Lo>RxPduId, <Lo>SduPtr)
<Up>_RxIndication(<Up>RxPduId,SduPtr)
Copy data from <Lo> Rx buffer
<Up>_RxIndication(<Up>RxPduId,SduPtr)
PduR_<Lo>IfRxIndication(<Lo>RxPduId, <Lo>SduPtr)
<Up> = {Com}
<Lo>If = {CanIf, FrIf, LinIf}

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.

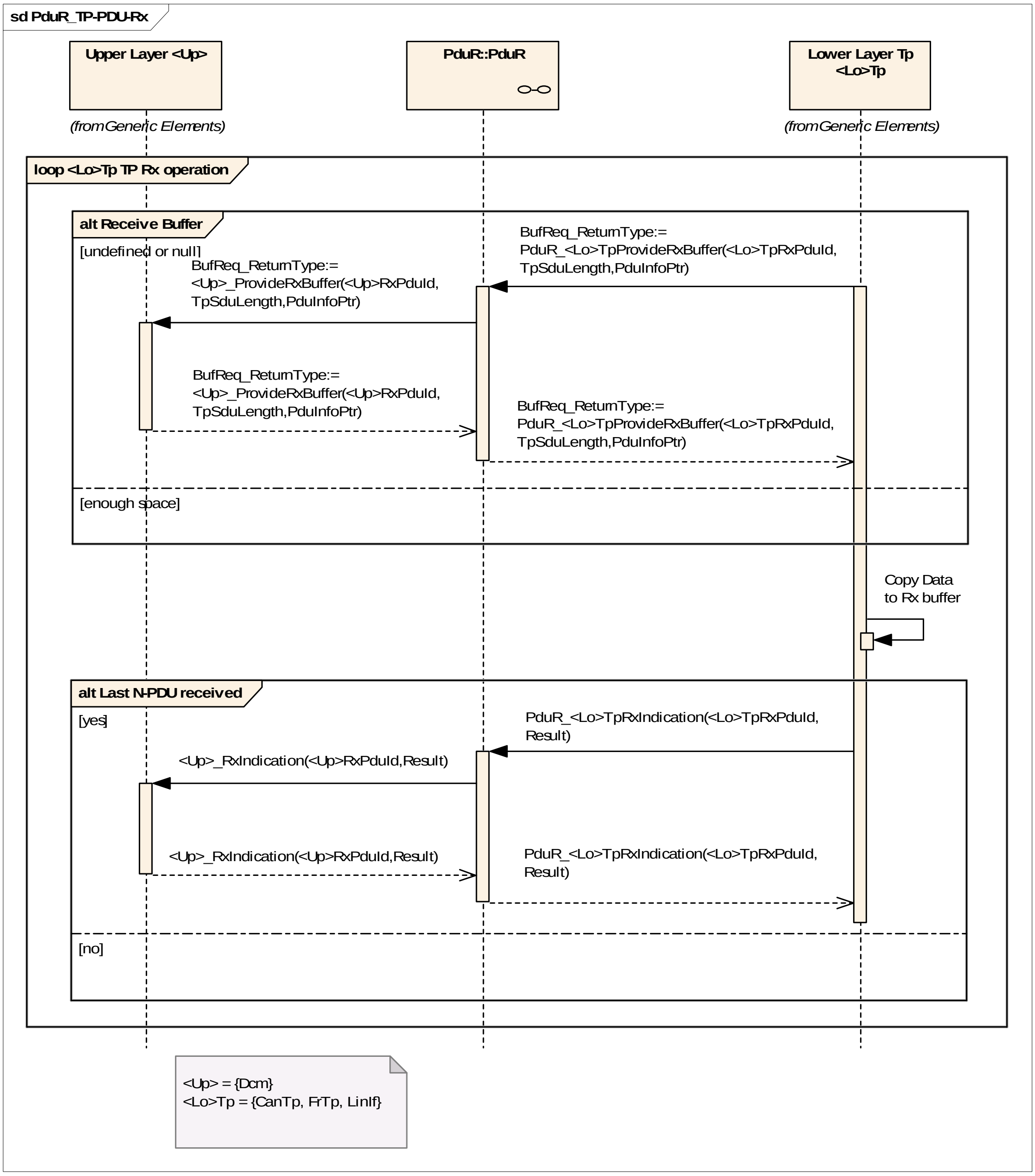

Figure 7: TP-PDU-Rx

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.

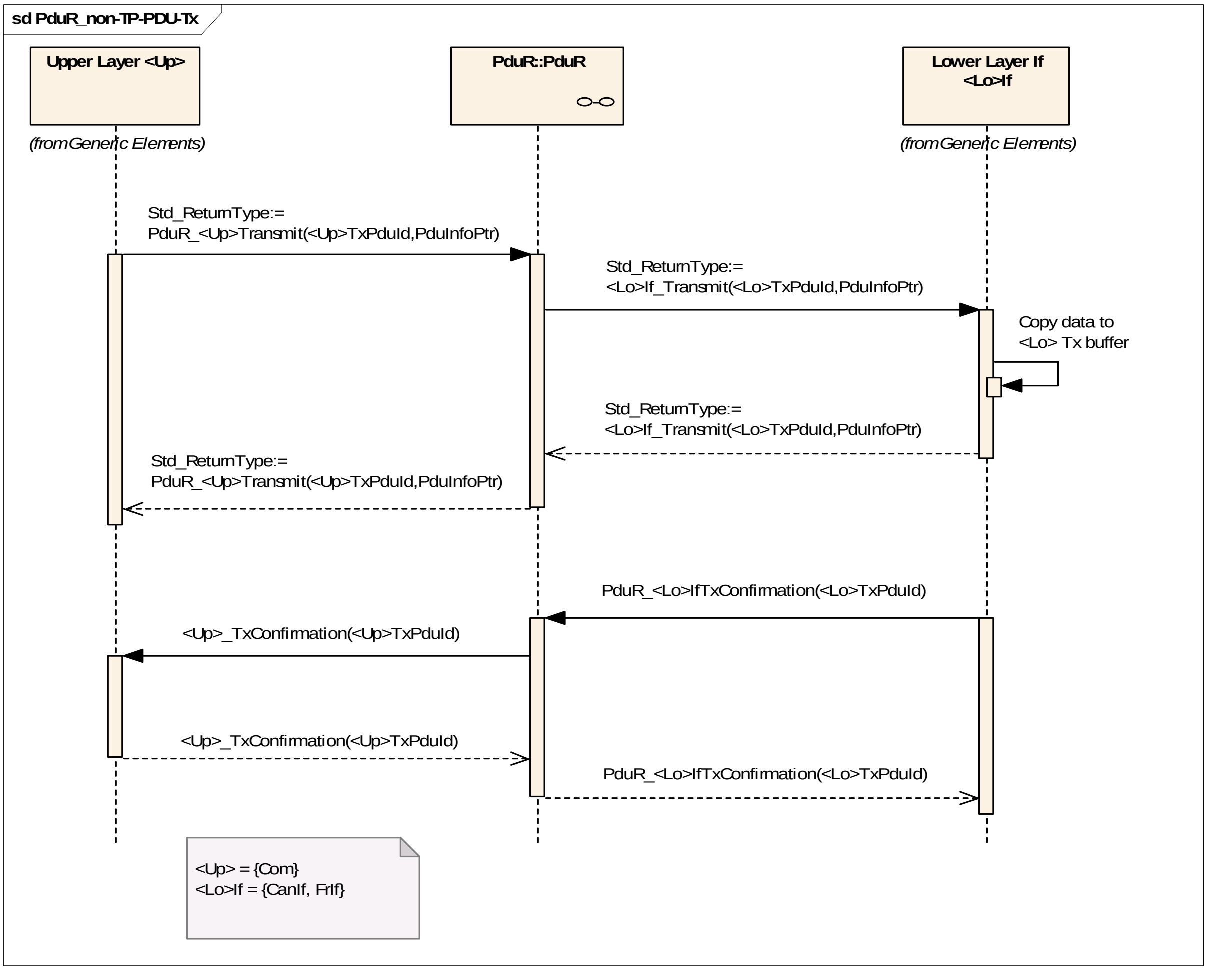
sd PduR_non-TP-PDU-Tx
Upper Layer <Up>
(fromGeneric Elements)
PduR::PduR
Lower Layer If <Lo>If
(fromGeneric Elements)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Copy data to <Lo> Tx buffer
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
PduR_<Lo>IfTxConfirmation(<Lo>TxPduId)
<Up>_TxConfirmation(<Up>TxPduId)
<Up>_TxConfirmation(<Up>TxPduId)
PduR_<Lo>IfTxConfirmation(<Lo>TxPduId)
<Up> = {Com}
<Lo>If = {CanIf, FrIf}

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.

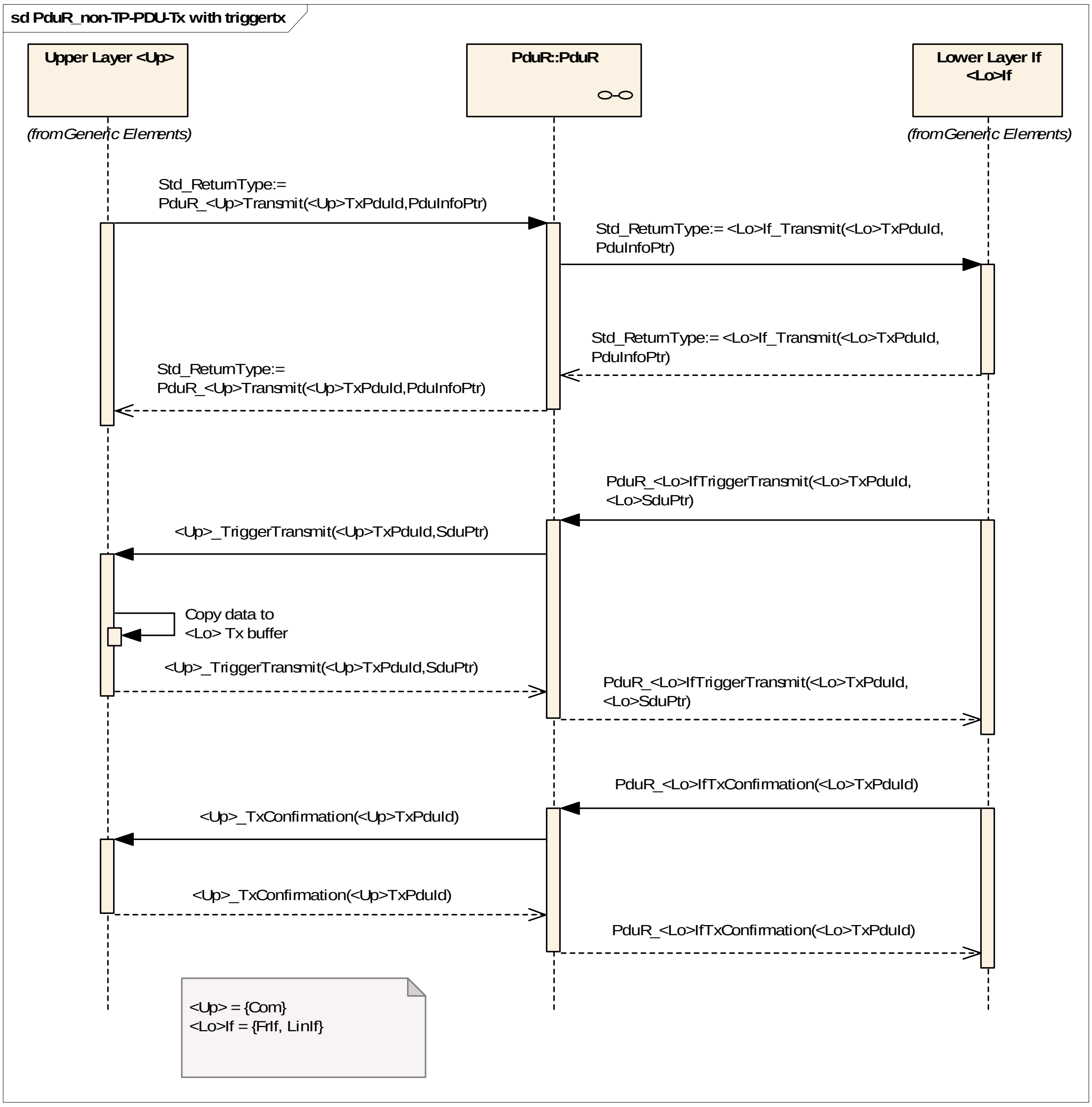
sd PduR_non-TP-PDU-Tx with triggertx
Upper Layer <Up>
PduR::PduR
Lower Layer If <Lo>If
(fromGeneric Elements)
(fromGeneric Elements)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId, PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId, PduInfoPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
<Up>_TriggerTransmit(<Up>TxPduId,SduPtr)
Copy data to <Lo> Tx buffer
<Up>_TriggerTransmit(<Up>TxPduId,SduPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
PduR_<Lo>IfTxConfirmation(<Lo>TxPduId)
<Up>_TxConfirmation(<Up>TxPduId)
<Up>_TxConfirmation(<Up>TxPduId)
PduR_<Lo>IfTxConfirmation(<Lo>TxPduId)
<Up> = {Com}
<Lo>If = {FrIf, LinIf}

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.

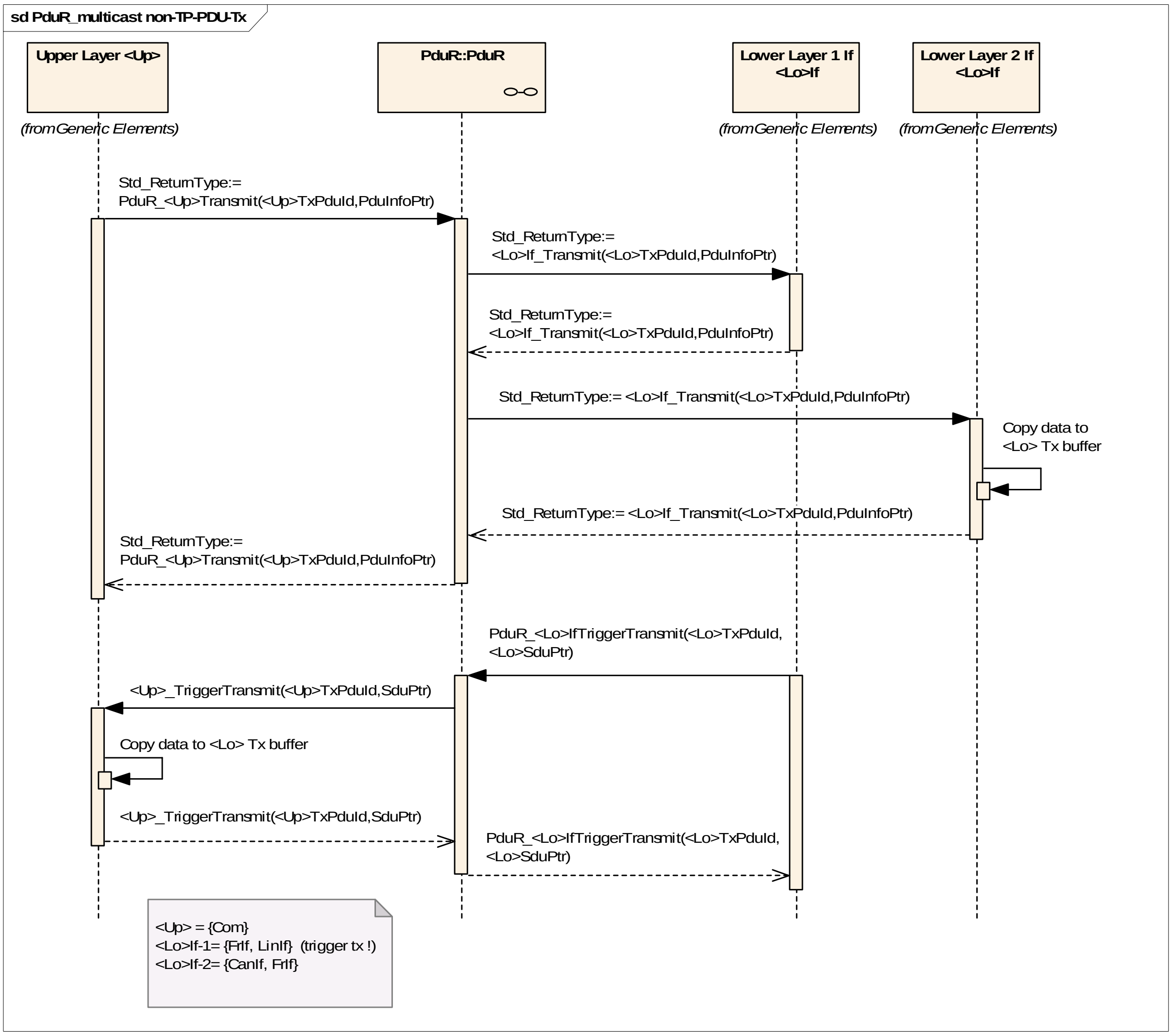
sd PduR_multicast non-TP-PDU-Tx
Upper Layer <Up>
PduR::PduR
Lower Layer 1 If <Lo>If
Lower Layer 2 If <Lo>If
(from Generic Elements)
(from Generic Elements)
(from Generic Elements)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Copy data to <Lo> Tx buffer
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
<Up>_TriggerTransmit(<Up>TxPduId,SduPtr)
Copy data to <Lo> Tx buffer
<Up>_TriggerTransmit(<Up>TxPduId,SduPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
<Up> = {Com}
<Lo>If-1= {FrIf, LinIf} (trigger tx !)
<Lo>If-2= {CanIf, FrIf}

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.

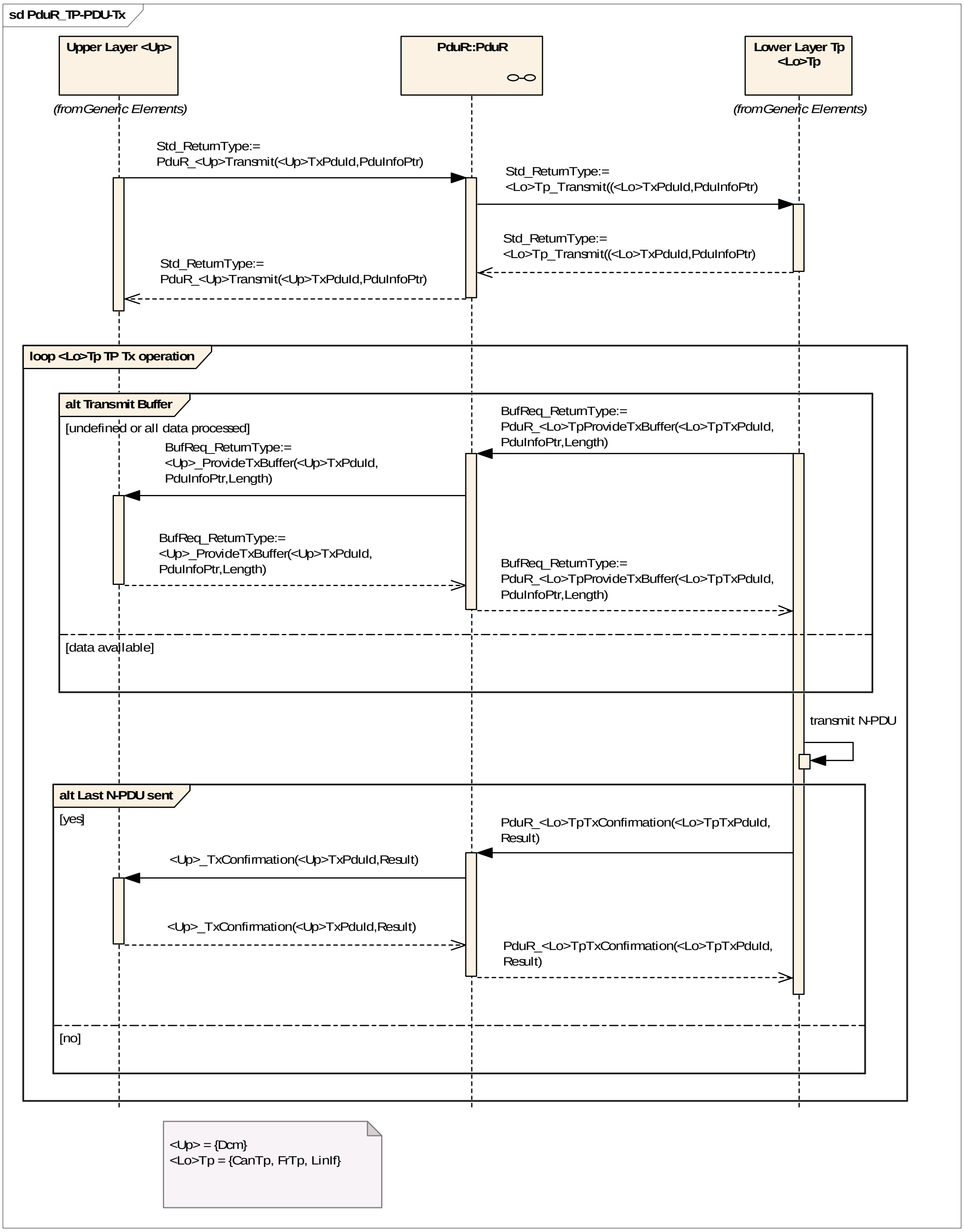
sd PduR_TP-PDU-Tx
Upper Layer <Up>
PduR::PduR
Lower Layer Tp <Lo>Tp
(fromGeneric Elements)
(fromGeneric Elements)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit((<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit((<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
loop <Lo>Tp TP Tx operation
alt Transmit Buffer
[undefined or all data processed]
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId, PduInfoPtr,Length)
BufReq_ReturnType:= <Up>_ProvideTxBuffer(<Up>TxPduId, PduInfoPtr,Length)
BufReq_ReturnType:= <Up>_ProvideTxBuffer(<Up>TxPduId, PduInfoPtr,Length)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId, PduInfoPtr,Length)
[data available]
transmit N-PDU
alt Last N-PDU sent
[yes]
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId, Result)
<Up>_TxConfirmation(<Up>TxPduId,Result)
<Up>_TxConfirmation(<Up>TxPduId,Result)
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId, Result)
[no]
<Up> = {Dcm}
<Lo>Tp = {CanTp, FrTp, LinIf}

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).

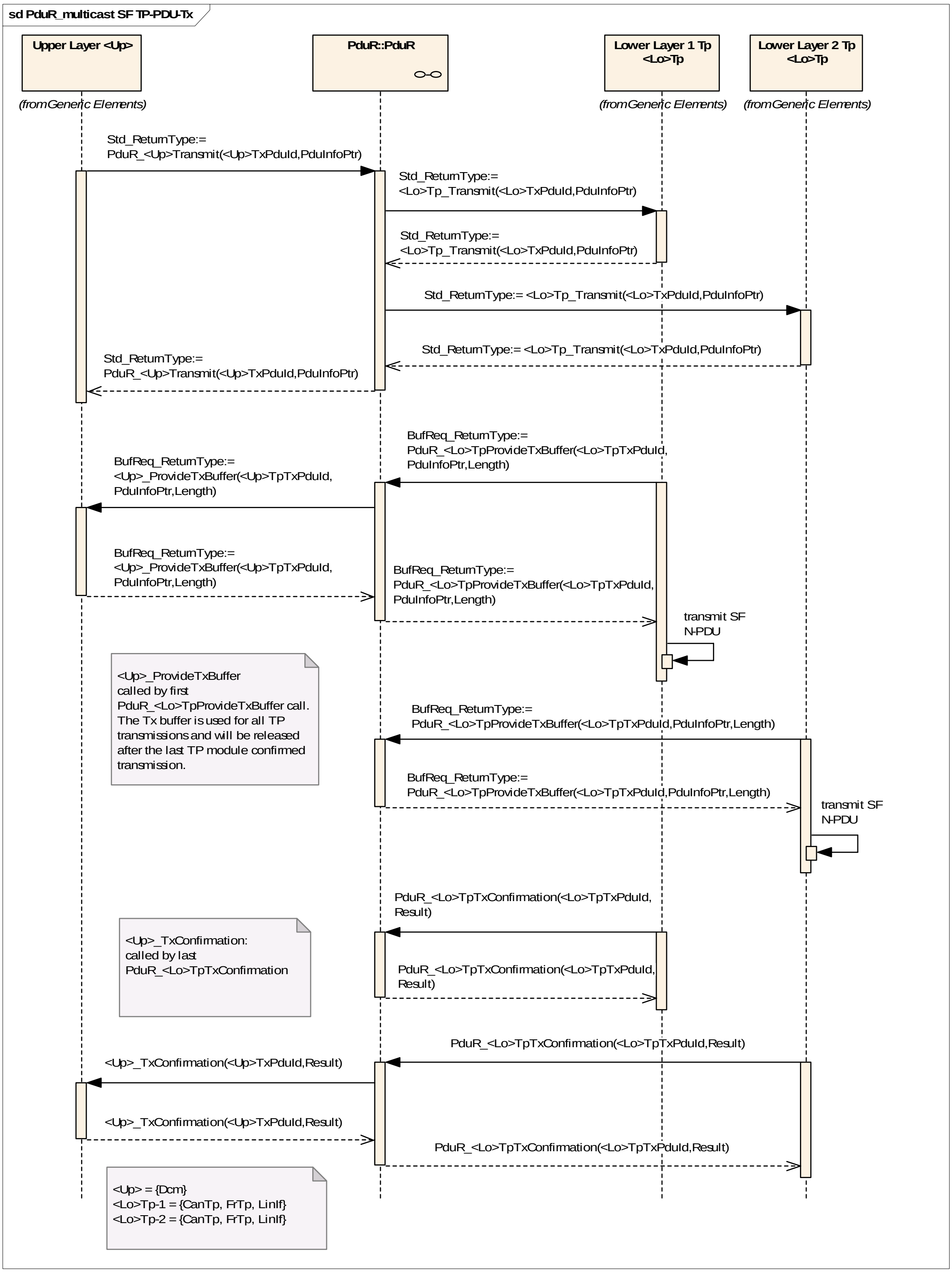
sd PduR_multicast SF TP-PDU-Tx
Upper Layer <Up>
(from Generic Elements)
PduR::PduR
Lower Layer 1 Tp <Lo>Tp
(from Generic Elements)
Lower Layer 2 Tp <Lo>Tp
(from Generic Elements)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId, PduInfoPtr,Length)
BufReq_ReturnType:= <Up>_ProvideTxBuffer(<Up>TpTxPduId, PduInfoPtr,Length)
BufReq_ReturnType:= <Up>_ProvideTxBuffer(<Up>TpTxPduId, PduInfoPtr,Length)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId, PduInfoPtr,Length)
transmit SF N-PDU
<Up>_ProvideTxBuffer called by first PduR_<Lo>TpProvideTxBuffer call. The Tx buffer is used for all TP transmissions and will be released after the last TP module confirmed transmission.
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId,PduInfoPtr,Length)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId,PduInfoPtr,Length)
transmit SF N-PDU
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId, Result)
<Up>_TxConfirmation: called by last PduR_<Lo>TpTxConfirmation
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId, Result)
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId,Result)
<Up>_TxConfirmation(<Up>TxPduId,Result)
<Up>_TxConfirmation(<Up>TxPduId,Result)
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId,Result)
<Up> = {Dcm}
<Lo>Tp-1 = {CanTp, FrTp, LinIf}
<Lo>Tp-2 = {CanTp, FrTp, LinIf}

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.

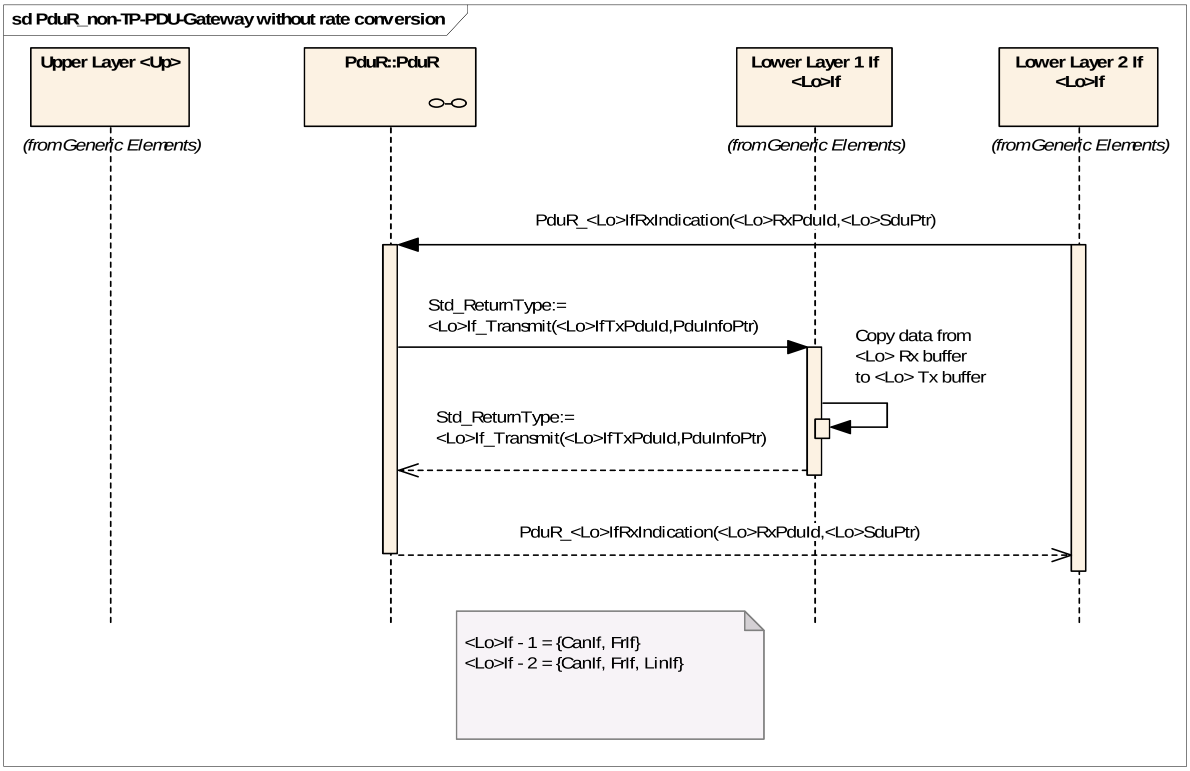
sd PduR_non-TP-PDU-Gateway without rate conversion
Upper Layer <Up>
(from Generic Elements)
PduR::PduR
Lower Layer 1 If
<Lo>If
(from Generic Elements)
Lower Layer 2 If
<Lo>If
(from Generic Elements)
PduR_<Lo>IfRxIndication(<Lo>RxPduId,<Lo>SduPtr)
Std_ReturnType:=
<Lo>If_Transmit(<Lo>IfTxPduId,PduInfoPtr)
Copy data from
<Lo> Rx buffer
to <Lo> Tx buffer
Std_ReturnType:=
<Lo>If_Transmit(<Lo>IfTxPduId,PduInfoPtr)
PduR_<Lo>IfRxIndication(<Lo>RxPduId,<Lo>SduPtr)
<Lo>If - 1 = {CanIf, FrIf}
<Lo>If - 2 = {CanIf, FrIf, LinIf}

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.

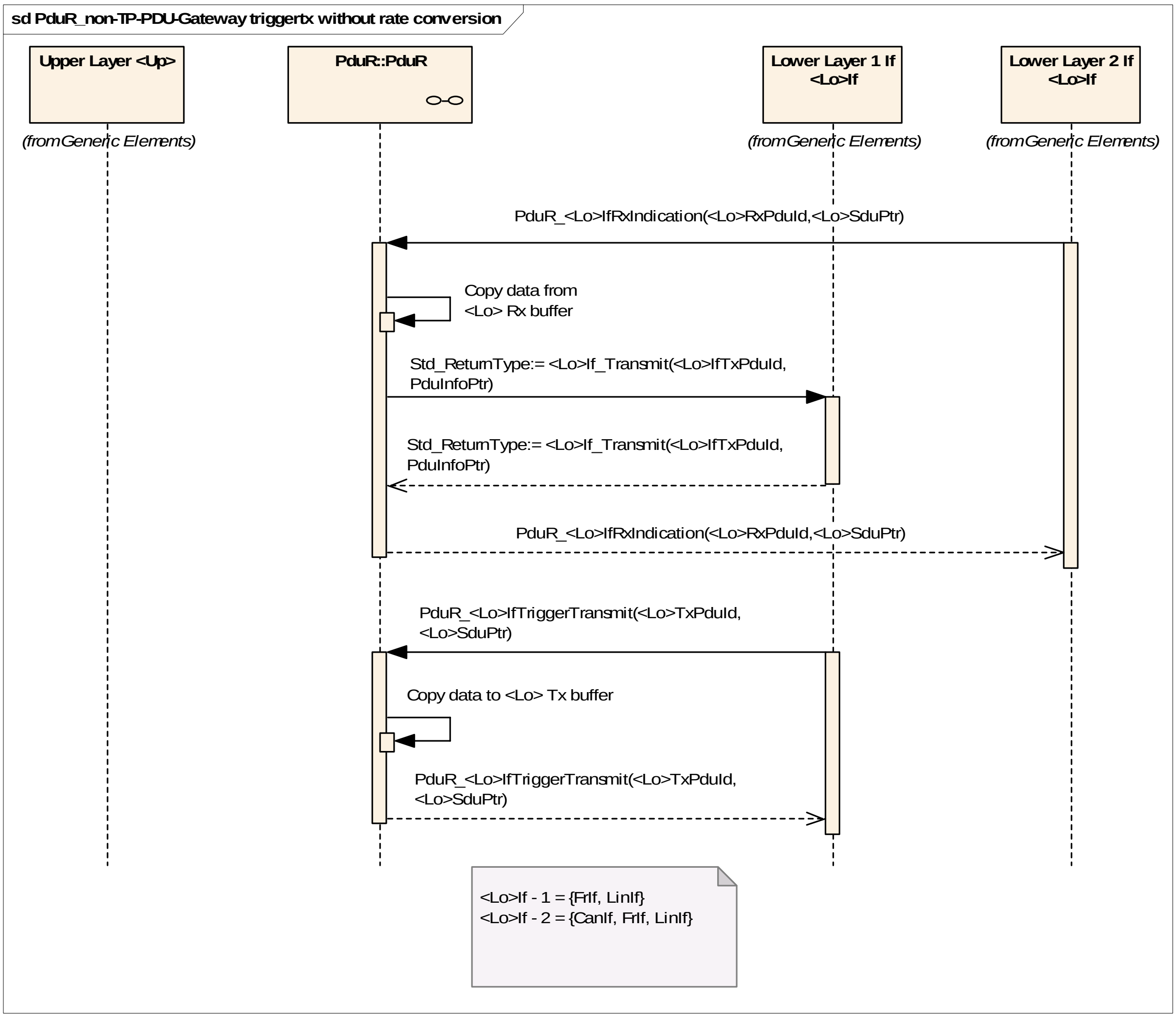
sd PduR_non-TP-PDU-Gateway triggertx without rate conversion
Upper Layer <Up>
PduR::PduR
Lower Layer 1 If <Lo>If
Lower Layer 2 If <Lo>If
(fromGeneric Elements)
(fromGeneric Elements)
(fromGeneric Elements)
PduR_<Lo>IfRxIndication(<Lo>RxPduId,<Lo>SduPtr)
Copy data from <Lo> Rx buffer
Std_ReturnType:= <Lo>If_Transmit(<Lo>IfTxPduId, PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>IfTxPduId, PduInfoPtr)
PduR_<Lo>IfRxIndication(<Lo>RxPduId,<Lo>SduPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
Copy data to <Lo> Tx buffer
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
<Lo>If - 1 = {FrIf, LinIf}
<Lo>If - 2 = {CanIf, FrIf, LinIf}

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).

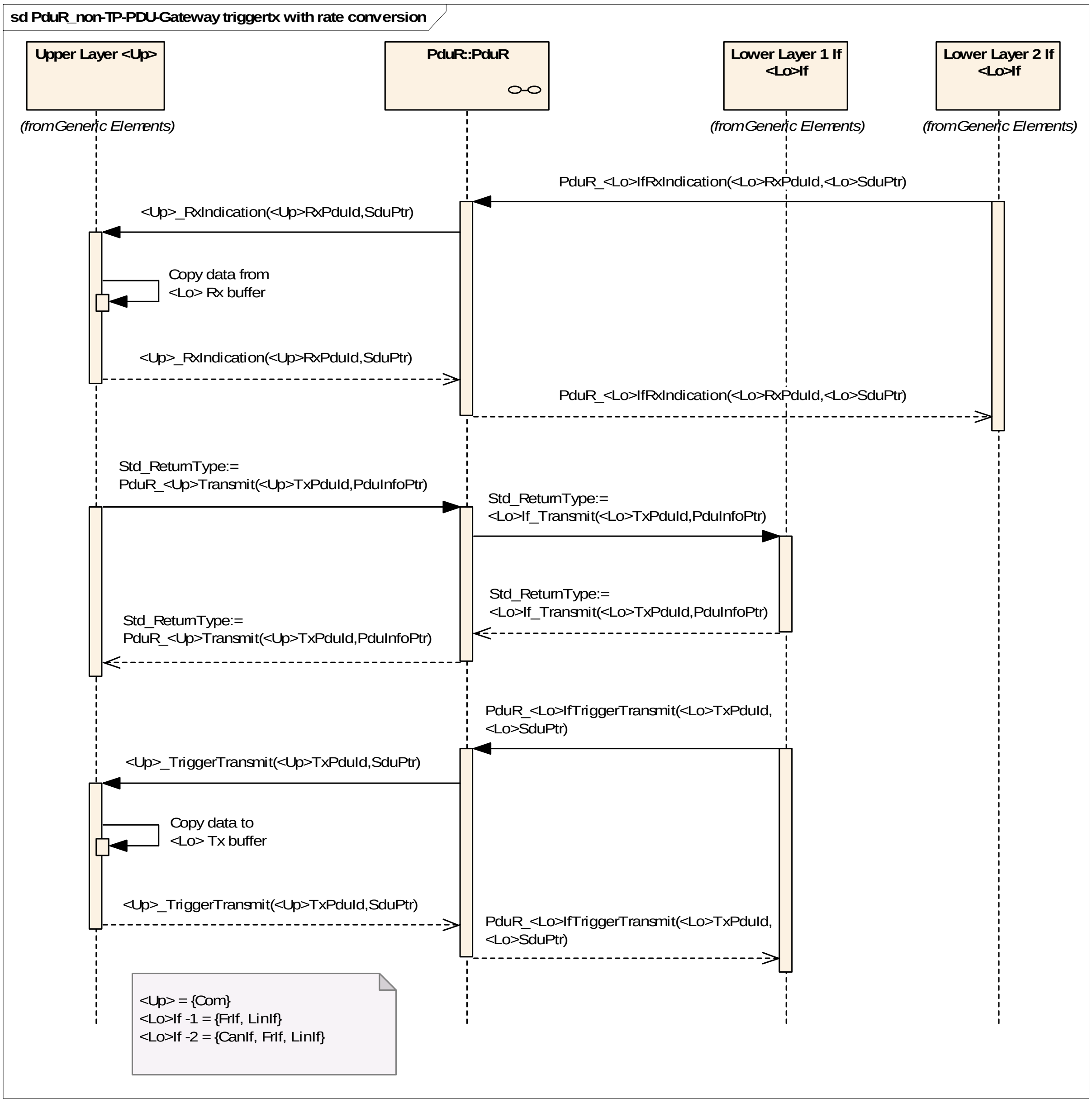
sd PduR_non-TP-PDU-Gateway triggertx with rate conversion
Upper Layer <Up>
(from Generic Elements)
PduR::PduR
Lower Layer 1 If <Lo>If
(from Generic Elements)
Lower Layer 2 If <Lo>If
(from Generic Elements)
PduR_<Lo>IfRxIndication(<Lo>RxPduId,<Lo>SduPtr)
<Up>_RxIndication(<Up>RxPduId,SduPtr)
Copy data from <Lo> Rx buffer
<Up>_RxIndication(<Up>RxPduId,SduPtr)
PduR_<Lo>IfRxIndication(<Lo>RxPduId,<Lo>SduPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>If_Transmit(<Lo>TxPduId,PduInfoPtr)
Std_ReturnType:= PduR_<Up>Transmit(<Up>TxPduId,PduInfoPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
<Up>_TriggerTransmit(<Up>TxPduId,SduPtr)
Copy data to <Lo> Tx buffer
<Up>_TriggerTransmit(<Up>TxPduId,SduPtr)
PduR_<Lo>IfTriggerTransmit(<Lo>TxPduId, <Lo>SduPtr)
<Up> = {Com}
<Lo>If -1 = {FrIf, LinIf}
<Lo>If -2 = {CanIf, FrIf, LinIf}

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).

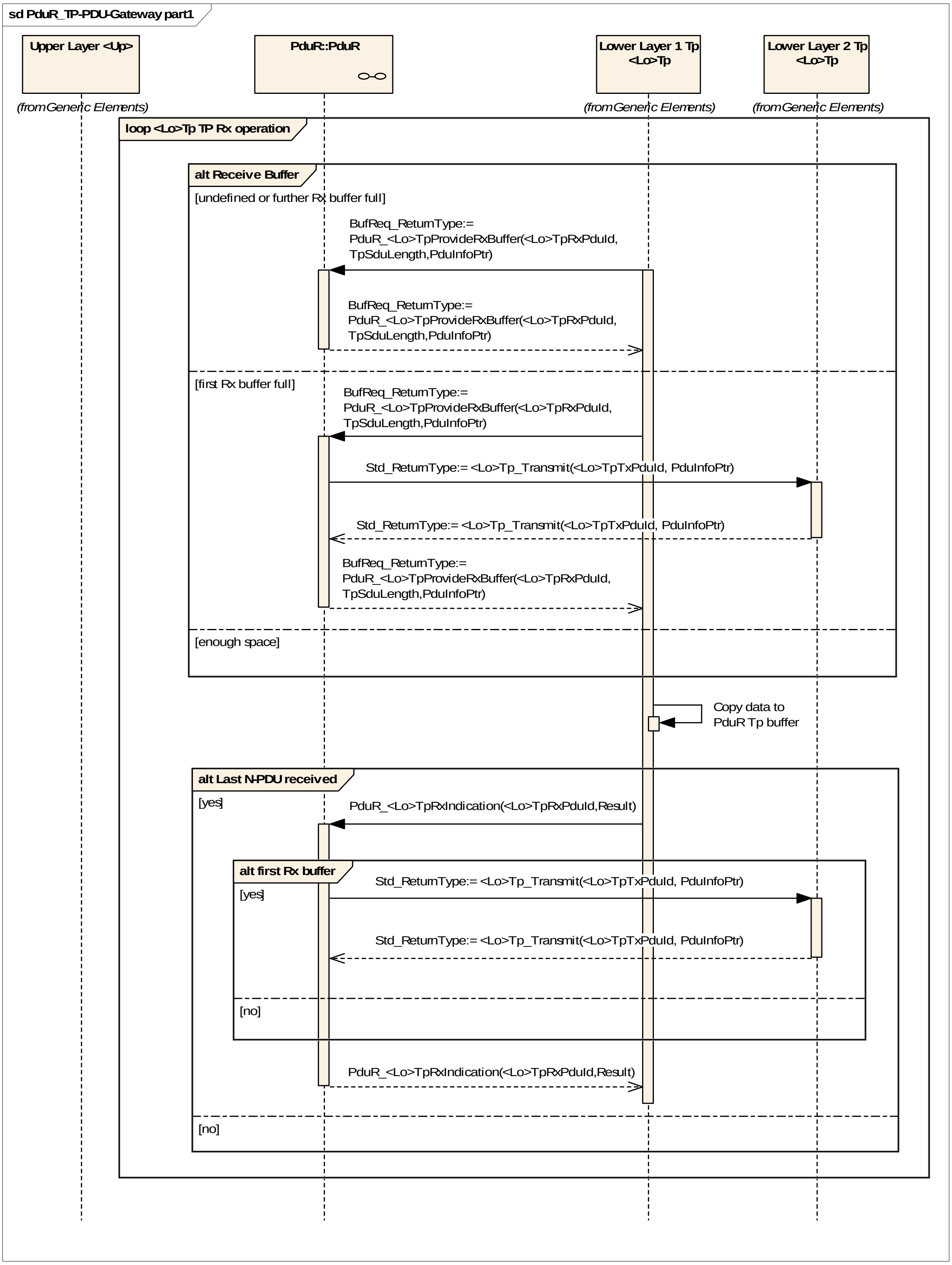
sd PduR_TP-PDU-Gateway part1
Upper Layer <Up>
PduR::PduR
Lower Layer 1 Tp <Lo>Tp
Lower Layer 2 Tp <Lo>Tp
(fromGeneric Elements)
(fromGeneric Elements)
(fromGeneric Elements)
loop <Lo>Tp TP Rx operation
alt Receive Buffer
[undefined or further Rx buffer full]
BufReq_ReturnType:= PduR_<Lo>TpProvideRxBuffer(<Lo>TpRxPduId, TpSduLength,PduInfoPtr)
BufReq_ReturnType:= PduR_<Lo>TpProvideRxBuffer(<Lo>TpRxPduId, TpSduLength,PduInfoPtr)
[first Rx buffer full]
BufReq_ReturnType:= PduR_<Lo>TpProvideRxBuffer(<Lo>TpRxPduId, TpSduLength,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId, PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId, PduInfoPtr)
BufReq_ReturnType:= PduR_<Lo>TpProvideRxBuffer(<Lo>TpRxPduId, TpSduLength,PduInfoPtr)
[enough space]
Copy data to PduR Tp buffer
alt Last N-PDU received
[yes]
PduR_<Lo>TpRxIndication(<Lo>TpRxPduId,Result)
alt first Rx buffer
[yes]
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId, PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId, PduInfoPtr)
[no]
PduR_<Lo>TpRxIndication(<Lo>TpRxPduId,Result)
[no]

Figure 16: TP PDU Gateway part 1

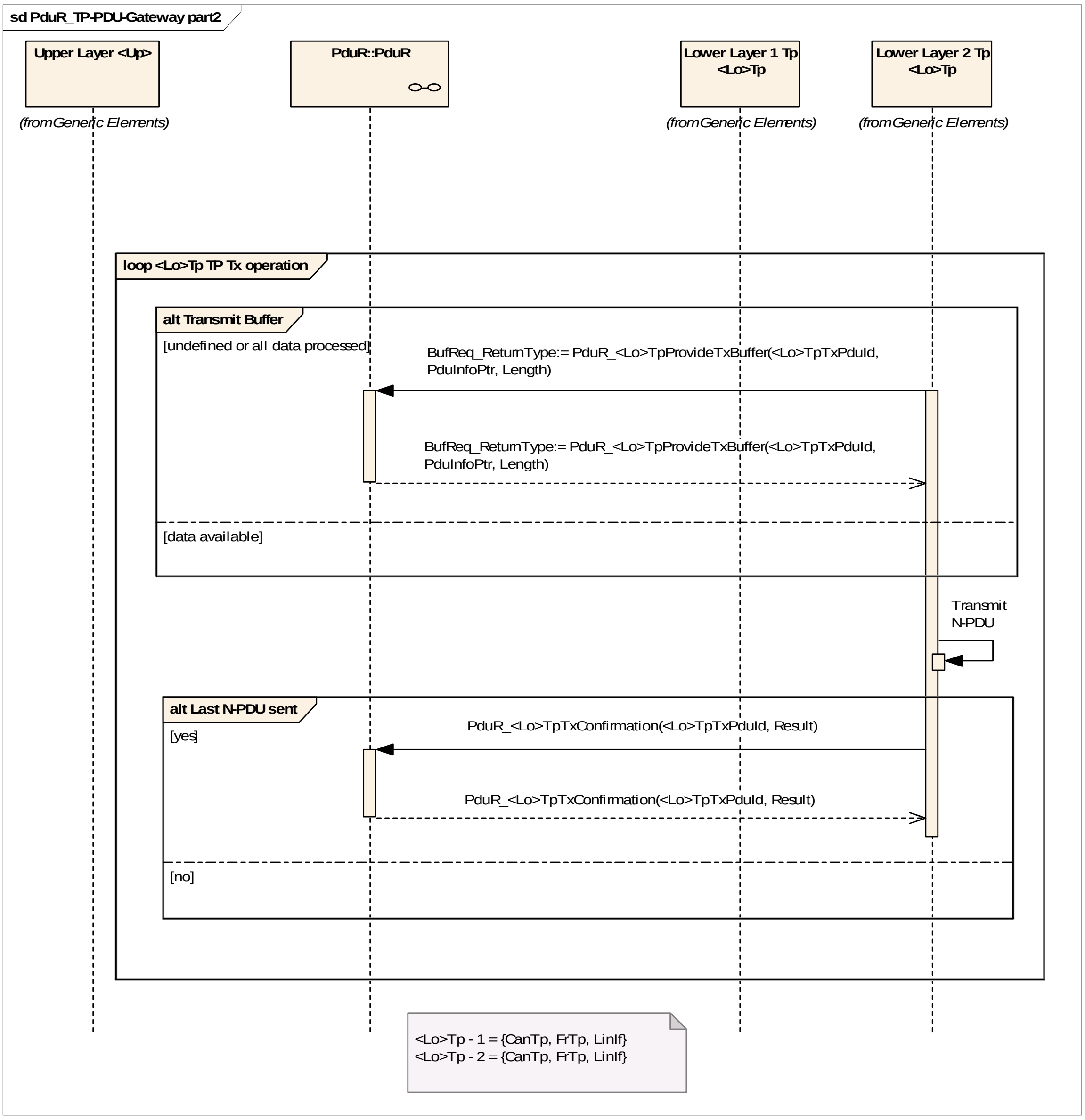
sd PduR_TP-PDU-Gateway part2
Upper Layer <Up>
(from Generic Elements)
PduR::PduR
Lower Layer 1 Tp <Lo>Tp
(from Generic Elements)
Lower Layer 2 Tp <Lo>Tp
(from Generic Elements)
loop <Lo>Tp TP Tx operation
alt Transmit Buffer
[undefined or all data processed]
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId, PduInfoPtr, Length)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId, PduInfoPtr, Length)
[data available]
Transmit N-PDU
alt Last N-PDU sent
[yes]
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId, Result)
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId, Result)
[no]
<Lo>Tp - 1 = {CanTp, FrTp, LinIf}
<Lo>Tp - 2 = {CanTp, FrTp, LinIf}

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.

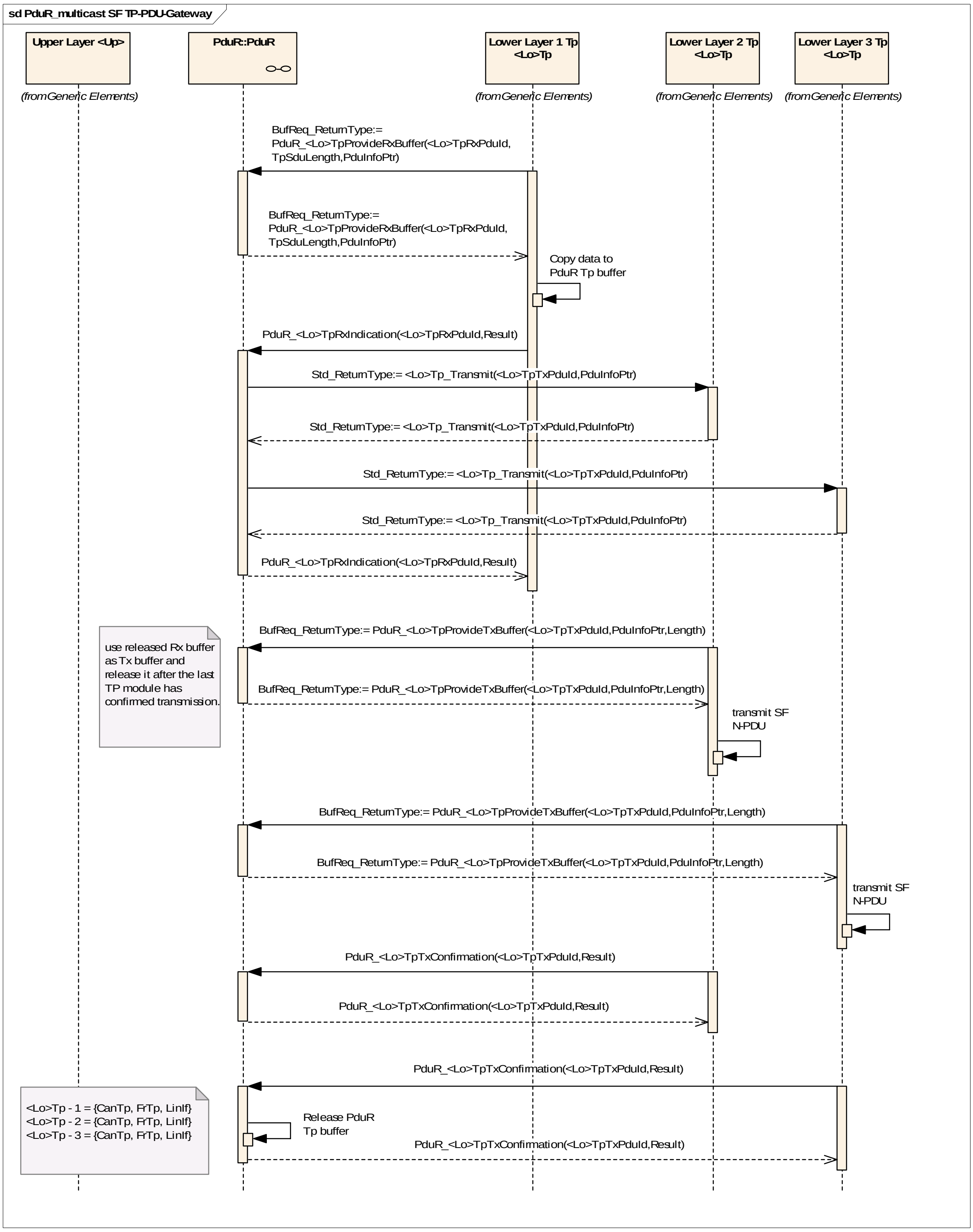
sd PduR_multicast SF TP-PDU-Gateway
Upper Layer <Up>
(fromGeneric Elements)
PduR::PduR
Lower Layer 1 Tp <Lo>Tp
(fromGeneric Elements)
Lower Layer 2 Tp <Lo>Tp
(fromGeneric Elements)
Lower Layer 3 Tp <Lo>Tp
(fromGeneric Elements)
BufReq_ReturnType:= PduR_<Lo>TpProvideRxBuffer(<Lo>TpRxPduId, TpSduLength,PduInfoPtr)
BufReq_ReturnType:= PduR_<Lo>TpProvideRxBuffer(<Lo>TpRxPduId, TpSduLength,PduInfoPtr)
Copy data to PduR Tp buffer
PduR_<Lo>TpRxIndication(<Lo>TpRxPduId,Result)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId,PduInfoPtr)
Std_ReturnType:= <Lo>Tp_Transmit(<Lo>TpTxPduId,PduInfoPtr)
PduR_<Lo>TpRxIndication(<Lo>TpRxPduId,Result)
use released Rx buffer as Tx buffer and release it after the last TP module has confirmed transmission.
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId,PduInfoPtr,Length)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId,PduInfoPtr,Length)
transmit SF N-PDU
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId,PduInfoPtr,Length)
BufReq_ReturnType:= PduR_<Lo>TpProvideTxBuffer(<Lo>TpTxPduId,PduInfoPtr,Length)
transmit SF N-PDU
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId,Result)
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId,Result)
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId,Result)
<Lo>Tp - 1 = {CanTp, FrTp, LinIf}
<Lo>Tp - 2 = {CanTp, FrTp, LinIf}
<Lo>Tp - 3 = {CanTp, FrTp, LinIf}
Release PduR Tp buffer
PduR_<Lo>TpTxConfirmation(<Lo>TpTxPduId,Result)

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

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.

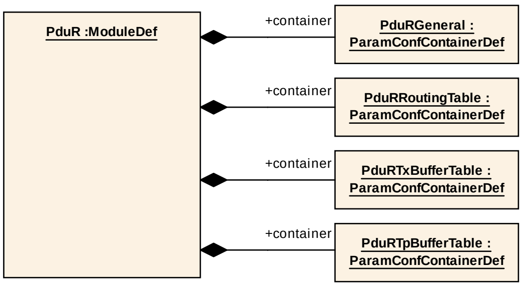
PduR :ModuleDef
+container
PduRGeneral : ParamConfContainerDef
+container
PduRRoutingTable : ParamConfContainerDef
+container
PduRTxBufferTable : ParamConfContainerDef
+container
PduRTpBufferTable : ParamConfContainerDef

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.

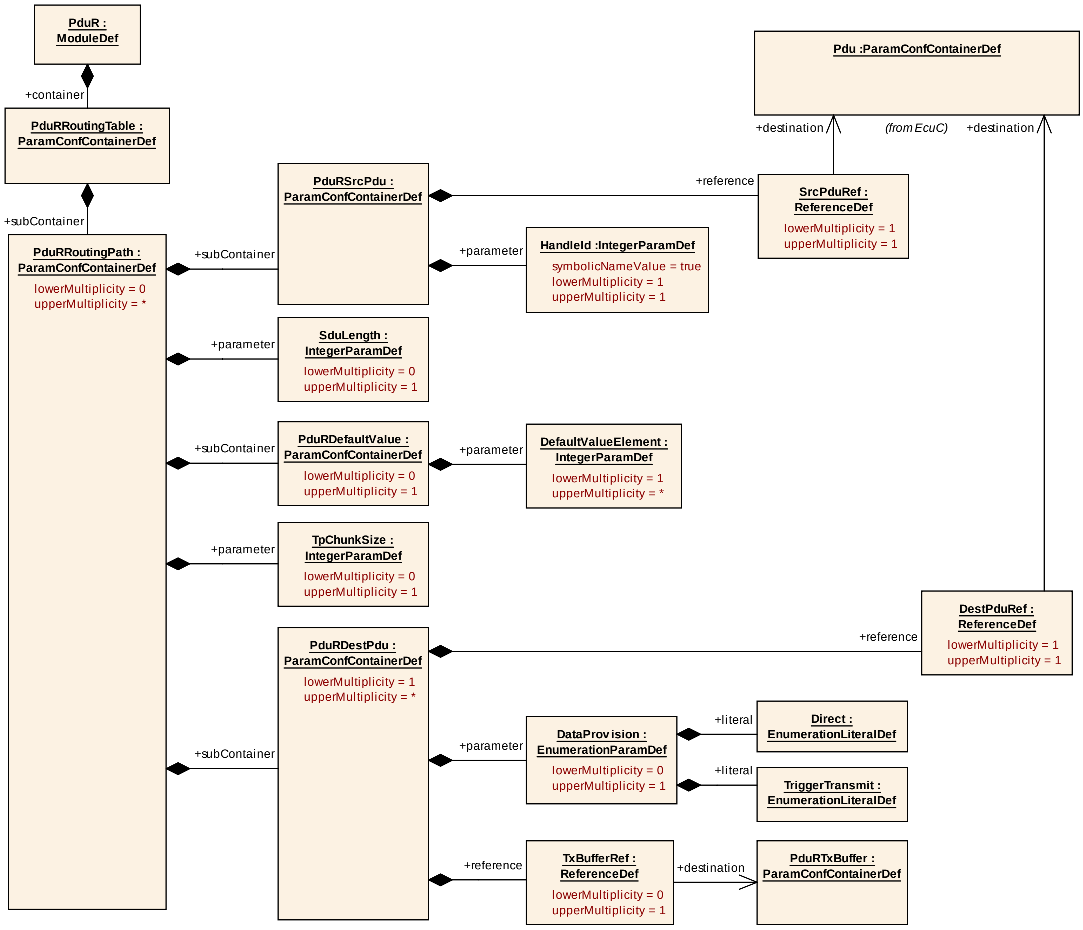

Figure 20: PDU Router Configuration Overview – Routing Paths

10.2.1 Variants

PDUR291: 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.

Variant 1: Only pre-compile time configuration parameters. This variant is only possible in zero-cost operation (i.e. all conditions stated in PDUR165 are fulfilled and pre-compile time configuration parameter PDUR_ZERO_COST_OPERATION is enabled).

Variant 2: A mix of pre-compile time and post-build time configuration parameters.

Variant 3: A mix of pre-compile time, link time and post-build time configuration parameters.

In fact only the configuration class of the minimum routing configuration parameters is different between Varaiant 2 and Variant 3, i.e. minimum routing is pre-compile time configurable in Variant 2 and link-time configurable in Variant 3.

10.2.2 PduR

SWS Item	**PDUR290**:
Container Name	PduR
Description	This container contains the configuration of the PDU Router.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRGeneral	1	module
PduRTxBufferTable	0..1	module / PduRGeneral/PDUR_GATEWAY_OPERATION
PduRTpBufferTable	0..1	module / PduRGeneral/PDUR_GATEWAY_OPERATION
PduRRoutingTable	1	module

10.2.3 PduRGeneral

SWS Item	**PDUR242**:
Container Name	PduRGeneral
Description	This container is a subcontainer of PduR and specifies the general configuration parameters of the PDU Router.
Configuration Parameters	

Name	PDUR_DEV_ERROR_DETECT
Description	Switches the Development Error Detection and Notification ON or OFF
Type	StringParamDef (#define)

Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	none		

Name	PDUR_VERSION_INFO_API		
Description	Activates/Deactivates the Version Info API (see chapter 8.3.1.2)		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	Version Info API activated	
	OFF	Version Info API deactivated	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	none		

Name	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.		
Type	IntegerParamDef (uint32)		
Unit	--		
Range	1	min	
	4294967295	max	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	PDUR_MEMORY_SIZE		
Description	Memory size reserved for PDU Router buffers. Only required for gateway operation.		
Type	IntegerParamDef (uint32)		
Unit	bytes		
Range	0	min	
	4294967295	max	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_GATEWAY_OPERATION		

Name	PDUR_ZERO_COST_OPERATION
Description	If all conditions stated in PDUR165 are fulfilled, 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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled (zero cost operation)	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	PDUR_SINGLE_IF		
Description	Single interface module in case zero cost operation is enabled (PDUR_ZERO_COST_OPERATION).		
Type	StringParamDef (#define)		
Unit	--		
Range	CanIf	Can interface	
	FrIf	FlexRay interface	
	LinIf	LIN interface	
Multiplicity	0 .. 1 (optional)		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	PDUR_SINGLE_TP		
Description	Single transport protocol module in case zero cost operation is enabled (PDUR_ZERO_COST_OPERATION).		
Type	StringParamDef (#define)		
Unit	--		
Range	CanTp	Can TP	
	FrTp	FlexRay TP	
	LinTp	LIN TP	
Multiplicity	0 .. 1 (optional)		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled (gateway operation)	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	PDUR_CANIF_SUPPORT

Description	Configuration parameter to enable or disable PDU Router support for CAN interface.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	PDUR_CANTP_SUPPORT		
Description	Configuration parameter to enable or disable PDU Router support for CAN TP.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	**--**		
Name	PDUR_FRIF_SUPPORT		
Description	Configuration parameter to enable or disable PDU Router support for FlexRay interface.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	PDUR_FRTP_SUPPORT		
Description	Configuration parameter to enable or disable PDU Router support for FlexRay TP.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	PDUR_LINIF_SUPPORT
Description	Configuration parameter to enable or disable PDU Router support for LIN interface.
Type	StringParamDef (#define)

Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	PDUR_LINTP_SUPPORT		
Description	Configuration parameter to enable or disable PDU Router support for LIN TP.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	PDUR_COM_SUPPORT		
Description	Configuration parameter to enable or disable PDU Router support for COM.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	PDUR_DCM_SUPPORT		
Description	Configuration parameter to enable or disable PDU Router support for DCM.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_GATEWAY_OPERATION		

Name	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.		
Type	StringParamDef (#define)		
Unit	--		
Range	ON	enabled	
	OFF	disabled	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	all Variants
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	PDUR_GATEWAY_OPERATION		

Name	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.		
Type	StringParamDef		
Unit	--		
Range	COM	COM	
	DCM	DCM	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	x	Variant 2
	Link time	x	Variant 3
	Post Build	--	--

Scope	module
Dependency	PDUR_ZERO_COST_OPERATION

Name	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.		
Type	StringParamDef		
Unit	--		
Range	CanIf	Can Interface	
	FrIf	FlexRay Interface	
	LinIf	LIN Interface	
	CanTp	Can TP	
	FrTp	FlexRay TP	
	LinTp	LIN TP	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	x	Variant 2
	Link time	x	Variant 3
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	IntegerParamDef (PduIdType)		
Unit	--		
Range	0	min	
	255/ 65535	max (depending on PduIdType)	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	x	Variant 2
	Link time	x	Variant 3
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	IntegerParamDef (PduIdType)		
Unit	--		
Range	0	min	
	255/ 65535	max (depending on PduIdType)	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	x	Variant 2
	Link time	x	Variant 3
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	IntegerParamDef (PduIdType)		
Unit	--		
Range	0	min	
	255/ 65535	max (depending on PduIdType)	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	x	Variant 2
	Link time	x	Variant 3
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

Name	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.		
Type	IntegerParamDef (PduIdType)		
Unit	--		
Range	0	min	
	255/ 65535	max (depending on PduIdType)	
Multiplicity	0..1 (optional)		
Configuration Class	***Pre-compile***	x	Variant 2
	Link time	x	Variant 3
	Post Build	--	--
Scope	module		
Dependency	PDUR_ZERO_COST_OPERATION		

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	

Name	PDUR_MAX_TX_BUFFER_NUMBER		
Description	maximum number of transmit buffers		
Type	IntegerParamDef (uint16)		
Unit	--		
Range	0	min	
	65535	max	
Multiplicity	1		
	Pre-compile	x	Variant 2, Variant 3

	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRTxBuffer	0..*	module

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	

Name	Length		
Description	Length of the buffer		
Type	IntegerParamDef (uint8)		
Unit	bytes		
Range	1	min	
	255	max	
Multiplicity	1		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	--		

Name	Depth		
Description	Specifies the depth of the buffer		
Type	IntegerParamDef (uint8)		
Unit	--		
Range	0	Single buffer	
	1	min FIFO depth	
	255	max FIFO depth	
Multiplicity	1		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	--		

10.2.6 PduRTpBufferTable

SWS Item	**PDUR245**:
Container Name	PduRTpBufferTable
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	

Name	PDUR_MAX_TP_BUFFER_NUMBER		
Description	maximum number of TP buffers		
Type	IntegerParamDef (uint16)		
Unit	--		
Range	0	min	
	65535	max	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	Variant 2, Variant 3
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRTpBuffer	0..*	module

10.2.7 PduRTpBuffer

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

Name	Length		
Description	Length of the buffer		
Type	IntegerParamDef (uint16)		
Unit	bytes		
Range	1	min	
	65535	max	
Multiplicity	1		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	--		

10.2.8 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..*	module

10.2.9 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	

Name	SduLength		
Description	Length of PDU data (SDU). Only required if a TX buffer is configured.		
Type	IntegerParamDef (uint8)		
Unit	byte		
Range	0	min	
	255	max	
Multiplicity	0 .. 1 (optional)		
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	PduRTxBufferTable/PduRTxBuffer/TxBuffer		

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.		
Type	IntegerParamDef (uint16)		
Unit	byte		
Range	1	min	
	65535	max	
Multiplicity	0 .. 1 (optional)		
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	PduRGeneral/PDUR_GATEWAY_OPERATION, PduRRoutingTable/PduRRoutingPath/PduRDestPdu, PduRTpBufferTable/PduRTpBuffer/Length		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
PduRSrcPdu	1	module
PduRDestPdu	1..*	module

PduRDefaultValue	0..1 (optional)	module / PduRGeneral/PDUR_GATEWAY_OPERATION and PduRRoutingTable/PduRRoutingPath/PduRDestPdu/DataProvision

10.2.10 PduRSrcPdu

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

Name	SrcPduRef		
Description	Source PDU reference; reference to unique PDU identifier which shall be used for the requested PDU Router operation.		
Type	ReferenceDef to Pdu		
Unit	--		
Range	--	--	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	--		

Name	HandleId		
Description	PDU identifier assigned by PDU Router		
Type	IntegerParamDef (PduIdType)		
Unit	--		
Range	0	min	
	255/ 65535	max	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	--		

10.2.11 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	

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.		
Type	ReferenceDef to Pdu		
Unit	--		
Range	--	--	
Multiplicity	1		
Configuration Class	***Pre-compile***	x	Variant 1
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	--		

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.		
Type	EnumerationParamDef		
Unit	--		
Range	Direct	direct data provision	
	TriggerTransmit	trigger transmit data provision	
Multiplicity	0 .. 1 (optional)		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	TxBufferRef (gateway PDUs with TriggerTransmit data provision require a TX buffer)		

Name	TxBufferRef		
Description	Specifies the assigned transmit buffer. Only required for specific non-TP gateway PDUs.		
Type	ReferenceDef		
Unit	--		
Range	--	--	
	--	--	
Multiplicity	0 .. 1 (optional)		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	referenced TxBuffer		

10.2.12 PduRDefaultValue

SWS Item	**PDUR289**:
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.
Configuration Parameters	

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 specifies the byte

	position of the element within the default value.		
Type	IntegerParamDef (uint8)		
Unit	--		
Range	0	min	
	255	max	
Multiplicity	1..*		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	L	Variant 2, Variant 3
Scope	module		
Dependency	PduRRoutingTable/PduRRoutingPath/SduLength		

10.3 Published Information

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

SWS Item	**PDUR236**:	
Information elements		
Information element name	***Type / Range***	***Information element description***
`PDUR_VENDOR_ID`	`#define / uint16`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`PDUR _MODULE_ID`	`#define / 0x33`	Module ID of this module from Module List
`PDUR_AR_MAJOR_VERSION`	`#define / uint8`	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`PDUR_AR_MINOR_VERSION`	`#define / uint8`	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`PDUR_AR_PATCH_VERSION`	`#define / uint8`	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`PDUR_SW_MAJOR_VERSION`	`#define / uint8`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`PDUR_SW_MINOR_VERSION`	`#define / uint8`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`PDUR_SW_PATCH_VERSION`	`#define / uint8`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

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 (sum of each TpBuffer length) must not exceed the reserved memory for PDU Router buffers specified by the pre-compile-time configuration parameter PDUR_MEMORY_SIZE.

(2) If the pre-compile time configuration parameter PDUR_ZERO_COST_OPERATION is enabled all conditions defined in PDUR165 must be fulfilled (e.g. PDUR_CANIF_SUPPORT and PDUR_FRIF_SUPPORT must not be enabled at the same time, PDUR_GATEWAY_OPERATION must not be enabled, ...).

(2) If PDUR_GATEWAY_OPERATION is disabled, the pre-compile time configuration parameters PDUR_SB_TX_BUFFER_SUPPORT, PDUR_FIFO_TX_BUFFER_SUPPORT, PDUR_MULTICAST_FROMIF_SUPPORT and PDUR_MULTICAST_FROMTP_SUPPORT must be disabled as well.

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***	***TargetFctPtr1***	***TargetFctPtr2***	***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_RxIndication	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>TpProvideTxBuffer 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***	***TargetFctPtr2***	***TargetFctPtr3***	***Target PduId***		
0	0	NULL	NULL	FrTp_Transmit	2		G
1	1	NULL	NULL	CanTp_Transmit	3	M	G
2	1	Dcm_ProvideRx Buffer	Dcm_RxIndication	NULL	2	M	
3	1	NULL	NULL	FrTp_Transmit	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 1

11.1 Deleted SWS Items

SWS Item	***Rationale***
PDUR156	Redundant with PDUR159
PDUR133	New SWS template – Configuration
PDUR205	Redundant with PDUR215
PDUR220	FIFO revised: TxConfirmation used instead of TriggerTransmit
PDUR146	Remark, not a requirement

11.2 Replaced SWS Items

SWS Item of Release 1	***replaced by SWS Item***	***Rationale***
PDUR118	PDUR227	New SWS template
PDUR173	PDUR236	New SWS template
PDUR157	PDUR236	New SWS template
PDUR219	PDUR240, PDUR241	New SWS template
PDUR212	PDUR255, PDUR258	FIFO revised

11.3 Changed SWS Items

SWS Item	***Rationale***
PDUR100	FIFO revised;development errors PDUR_E_PDU_ID_BUSY, PDUR_E_IF_TX_REQ_REJECTED and PDUR_E_IF_TX_CONF_UNUSED removed; development error PDUR_E_CONFIG_PARAM renamed to PDUR_E_CONFIG_PTR_INVALID; development error values renumbered; production error PDUR_E_INIT_FAILED added
PDUR101	New SWS template
PDUR102	Reference to chapter 10 (Configuration specification) added
PDUR103	New SWS template, clarification
PDUR165	Clarification, IPDUM added
PDUR201	Clarification, COM is limited to communication via interface modules
PDUR202	Clarification, DCM is limited to communication via TP modules.
PDUR225	Adapted to extended configuration; zero cost operation
PDUR134	moved to chapter 7.1; clarification
PDUR216	IPDUM
PDUR132	IPDUM, <module>.h
PDUR166	Clarification
PDUR168	Clarification
PDUR142	Clarification
PDUR170	Clarification
PDUR211	FIFO revised
PDUR214	Usage of exclusive areas
PDUR172	Bugfix, clarification
PDUR178	FIFO revised
PDUR108	PDU transmit buffer handling (clarification, revised FIFO), zero cost operation
PDUR193	PDU transmit buffer handling (clarification, revised FIFO)
PDUR194	PDU transmit buffer handling (clarification, revised FIFO)

PDUR195	PDU transmit buffer handling (clarification, revised FIFO)
PDUR196	PDU transmit buffer handling (clarification, revised FIFO)
PDUR199	PDU transmit buffer handling (clarification, revised FIFO)
PDUR197	PDU transmit buffer handling (clarification, revised FIFO)
PDUR198	PDU transmit buffer handling (clarification, revised FIFO)
PDUR200	PDU transmit buffer handling (clarification, revised FIFO)
PDUR224	Clarification
PDUR215	Clarification
PDUR174	Clarification
PDUR203	Clarification
PDUR221	Clarification
PDUR106	Production error PDUR_E_INIT_FAILED added
PDUR208	Clarification
PDUR222	development error PDUR_E_CONFIG_PARAM renamed to PDUR_E_CONFIG_PTR_INVALID

11.4 Added SWS Items

SWS Item	***Rationale***
PDUR227	Clarification regarding API parameter checking; new SWS template
PDUR231	New SWS template
PDUR232	New SWS template
PDUR233	Clarification regarding production error detection; new SWS template
PDUR234	PduR_GetVersionInfo API added; new SWS template
PDUR235	PduR_GetVersionInfo API added; new SWS template
PDUR236	New SWS template
PDUR237	PduR_IpdumTransmit API added
PDUR238	PduR_IpdumTransmit API added
PDUR239	Clarification
PDUR240	New SWS template
PDUR241	New SWS template
PDUR242	New SWS template – Configuration
PDUR243	New SWS template – Configuration
PDUR244	New SWS template – Configuration
PDUR245	New SWS template – Configuration
PDUR246	New SWS template – Configuration
PDUR247	New SWS template – Configuration
PDUR248	New SWS template – Configuration
PDUR249	New SWS template – Configuration
PDUR250	Implementation of BSW171
PDUR251	Clarification; IPDUM added
PDUR252	FIFO revised
PDUR253	FIFO revised
PDUR254	FIFO revised
PDUR255	FIFO revised
PDUR256	FIFO revised
PDUR257	FIFO revised
PDUR258	FIFO revised
PDUR259	FIFO revised
PDUR260	Clarification regarding PDU transmit buffers which are configured as single buffers
PDUR280	PduR_GetConfigurationId API added
PDUR281	Configuration identifier
PDUR284	PduR_StateType (definition without SWS item ID)
PDUR285	minimum routing
PDUR286	minimum routing

PDUR287	zero cost operation
PDUR288	Configuration
PDUR289	Configuraiton
PDUR290	Configuration
PDUR291	Configuration variants