

<b>Document Title</b>	Specification of PDU Router
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	035
<b>Document Classification</b>	Standard

<b>Document Version</b>	2.3.0
<b>Document Status</b>	Final
<b>Part of Release</b>	3.1
<b>Revision</b>	5

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
20.09.2010	2.3.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Added support for Gatewaying longer I-PDUs that configured</li> <li>Added return type to TriggerTransmit APIs</li> <li>Changed to PduInfoType in RxIndication</li> </ul>
23.06.2008	2.2.2	AUTOSAR Administration	Legal disclaimer revised
22.01.2008	2.2.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Correction figure 3</li> </ul>
13.11.2007	2.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Variants have been renamed.</li> <li>New Callbacks PduR_LinTpChangeParameterConfirmation, PduR_FrTpChangeParameterConfirmation</li> <li>PduR_CanTpChangeParameterConfirmation has been added.</li> <li>New API's PduR_ChangeParameterRequest, PduR_CancelTransmitRequest has been added</li> <li>New Typedefines PduR_ParameterValueType, PduR_CancelReasonType has been added</li> <li>Document meta information extended</li> <li>Small layout adaptations made</li> </ul>

31.01.2007	2.1.0	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Clarifications added (FIFO, TxConf, ...)</li><li>• Unnecessary development errors removed</li><li>• SchM_PduR.h and MemMap.h added</li><li>• Corrections of configuration parameters</li><li>• More details in Chapter 11</li> <li>• Legal disclaimer revised</li><li>• Release Notes added</li><li>• “Advice for users” revised</li><li>• “Revision Information” added</li></ul>
28.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template. <ul style="list-style-type: none"><li>• Major changes in chapter 10</li><li>• Structure of document changed partly</li><li>• Other changes see chapter</li></ul>
20.06.2005	1.0.0	AUTOSAR Administration	Initial Release

## Disclaimer

This specification and the material contained in it, as released by AUTOSAR is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only.  
For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Advice for users

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

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

## Table of Contents

1	Introduction and functional overview .....	7
2	Acronyms and abbreviations .....	10
3	Related documentation.....	11
3.1	Input documents.....	11
3.2	Related standards and norms .....	12
4	Constraints and assumptions .....	13
4.1	Limitations .....	13
4.2	Applicability to car domains.....	13
5	Dependencies to other modules.....	14
5.1	File structure .....	14
5.1.1	Code file structure .....	14
5.1.2	Header file structure.....	14
6	Requirements traceability .....	17
7	Functional Specification.....	21
7.1	General Behavior .....	21
7.1.1	PDU Reception .....	22
7.1.2	PDU Transmission .....	23
7.1.3	PDU Gateway .....	24
7.2	Zero Cost Operation.....	28
7.3	Minimum Routing .....	28
7.4	Reentrancy of API calls .....	29
7.5	State Management.....	29
7.6	Error classification .....	30
7.7	Error detection.....	31
7.8	Error notification .....	31
7.9	API parameter checking .....	32
8	API specification.....	33
8.1	Imported types.....	33
8.2	Type definitions .....	33
8.2.1	PduR_StateType.....	33
8.2.2	PduR_LConfigType .....	33
8.2.3	PduR_PBConfigType .....	34
8.2.4	PduR_CancelReasonType.....	34
8.2.5	PduR_ParameterValueType .....	34
8.3	Function definitions .....	35
8.3.1	General functions provided by the PDU Router .....	35
8.3.1.1	PduR_Init .....	35
8.3.1.2	PduR_GetVersionInfo .....	36
8.3.1.3	PduR_GetConfigurationId .....	36
8.3.1.4	PduR_CancelTransmitRequest.....	37
8.3.1.5	PduR_ChangeParameterRequest: .....	38
8.3.2	Function definitions for CAN interaction .....	38

8.3.2.1	PduR_CanIfRxIndication .....	38
8.3.2.2	PduR_CanIfTxConfirmation .....	39
8.3.2.3	PduR_CanTpProvideRxBuffer .....	40
8.3.2.4	PduR_CanTpRxIndication.....	41
8.3.2.5	PduR_CanTpProvideTxBuffer.....	42
8.3.2.6	PduR_CanTpTxConfirmation .....	44
8.3.2.7	PduR_CanTpChangeParameterConfirmation.....	45
8.3.3	Function definitions for FlexRay interaction .....	46
8.3.3.1	PduR_FrlfRxIndication .....	46
8.3.3.2	PduR_FrlfTxConfirmation .....	47
8.3.3.3	PduR_FrlfTriggerTransmit .....	48
8.3.3.4	PduR_FrTpProvideRxBuffer .....	48
8.3.3.5	PduR_FrTpRxIndication.....	50
8.3.3.6	PduR_FrTpProvideTxBuffer.....	51
8.3.3.7	PduR_FrTpTxConfirmation .....	52
8.3.3.8	PduR_FrTpChangeParameterConfirmation .....	54
8.3.4	Function definitions for LIN interaction .....	55
8.3.4.1	PduR_LinIfRxIndication.....	55
8.3.4.2	PduR_LinIfTxConfirmation .....	56
8.3.4.3	PduR_LinIfTriggerTransmit .....	57
8.3.4.4	PduR_LinTpProvideRxBuffer .....	58
8.3.4.5	PduR_LinTpRxIndication .....	59
8.3.4.6	PduR_LinTpProvideTxBuffer .....	60
8.3.4.7	PduR_LinTpTxConfirmation.....	62
8.3.4.8	PduR_LinTpChangeParameterConfirmation.....	63
8.3.5	Function definitions for COM interaction .....	65
8.3.5.1	PduR_ComTransmit .....	65
8.3.6	Function definitions for DCM interaction .....	66
8.3.6.1	PduR_DcmTransmit.....	66
8.3.7	Function definitions for IPDUM interaction .....	68
8.3.7.1	PduR_IpdumTransmit .....	68
8.3.7.2	PduR_IpdumTxConfirmation.....	68
8.3.7.3	PduR_IpdumRxIndication .....	69
8.4	Scheduled functions .....	70
8.5	Expected Interfaces.....	70
8.5.1	Mandatory Interfaces .....	70
8.5.2	Optional Interfaces .....	70
8.5.3	Configurable interfaces .....	71
9	Sequence diagrams .....	72
9.1	Initialization .....	72
9.2	PDU Reception .....	73
9.3	PDU Transmission .....	74
9.4	PDU Gateway .....	80
10	Configuration specification.....	89
10.1	How to read this chapter .....	89
10.1.1	Configuration and configuration parameters .....	89
10.1.2	Variants.....	89
10.1.3	Containers.....	90
10.1.4	Specification template for configuration parameters .....	90

10.2	Containers and configuration parameters .....	91
10.2.1	Variants .....	92
10.2.2	PduR .....	93
10.2.3	PduRGeneral .....	93
10.2.4	PduRTxBufferTable .....	100
10.2.5	PduRTxBuffer .....	101
10.2.6	PduRRoutingTable .....	101
10.2.7	PduRRoutingPath .....	101
10.2.8	PduRSrcPdu .....	102
10.2.9	PduRDestPdu .....	103
10.2.10	PduRDefaultValue .....	104
10.2.11	PduRDefaultValueElement .....	104
10.2.12	PduRTpBufferTable .....	105
10.2.13	PduRTpBuffer .....	105
10.2.14	PduRGlobalConfig .....	106
10.3	Published Information .....	106
10.4	Plausibility checks of configuration .....	108
10.5	Example structure of Routing tables .....	108
10.5.1	Routing tables for communication via interface modules .....	108
10.5.2	Routing tables for communication via transport protocol modules ....	111
11	Changes to Release 2 .....	113
11.1	Deleted SWS Items .....	113
11.2	Replaced SWS Items .....	113
11.3	Changed SWS Items .....	113
11.4	Added SWS Items .....	114
12	Changes during SWS Improvements by Technical Office .....	115
12.1	Deleted SWS Items .....	115
12.2	Replaced SWS Items .....	115
12.3	Changed SWS Items .....	116
12.4	Added SWS Items .....	116

## 1 Introduction and functional overview

This specification describes the functionality and API for the AUTOSAR PDU Router module.

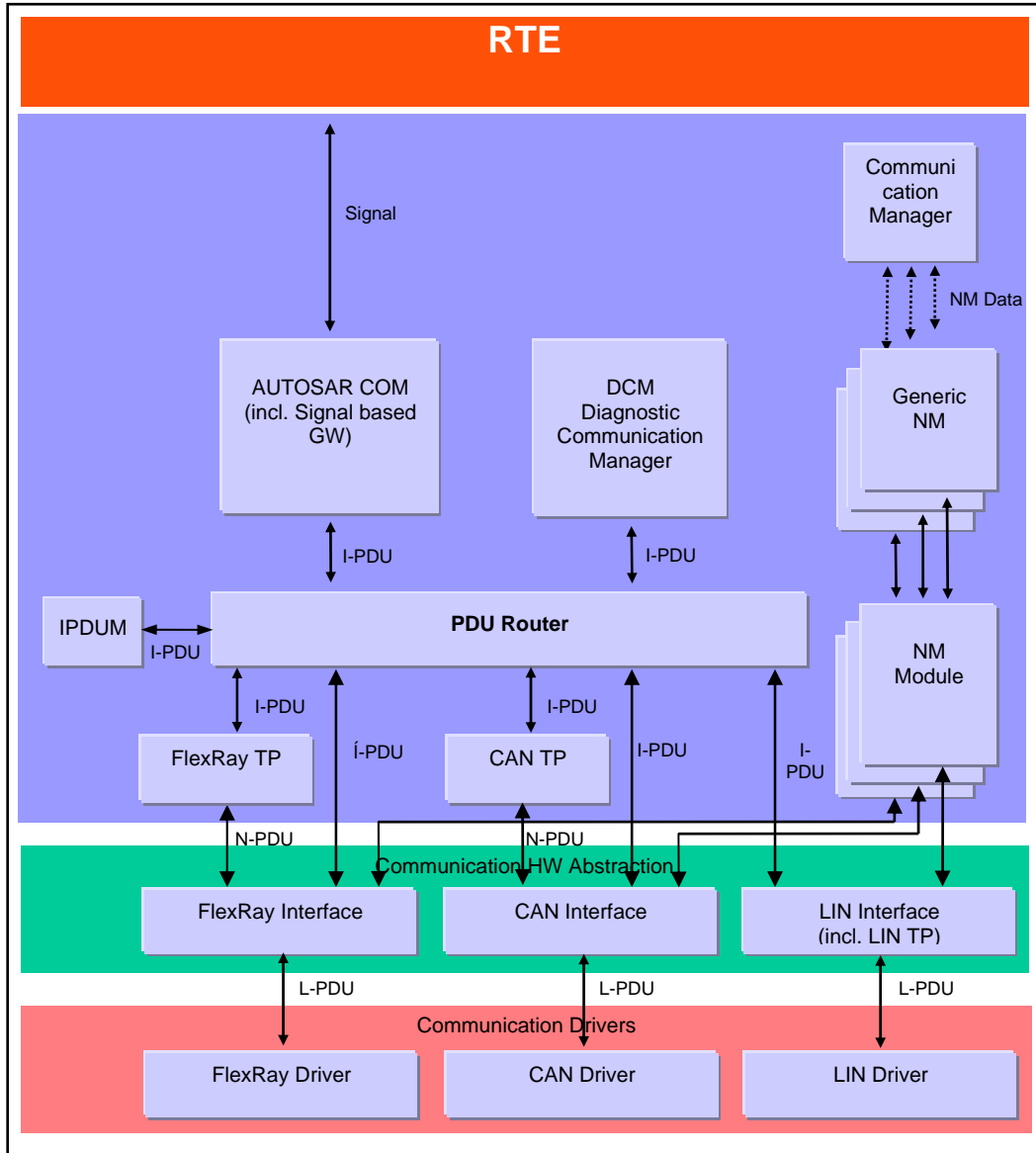
The PDU Router module provides services for routing of I-PDUs (Interaction Layer Protocol Data Units) between the following modules:

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

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

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

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



**Figure 1: Communication Structure**

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

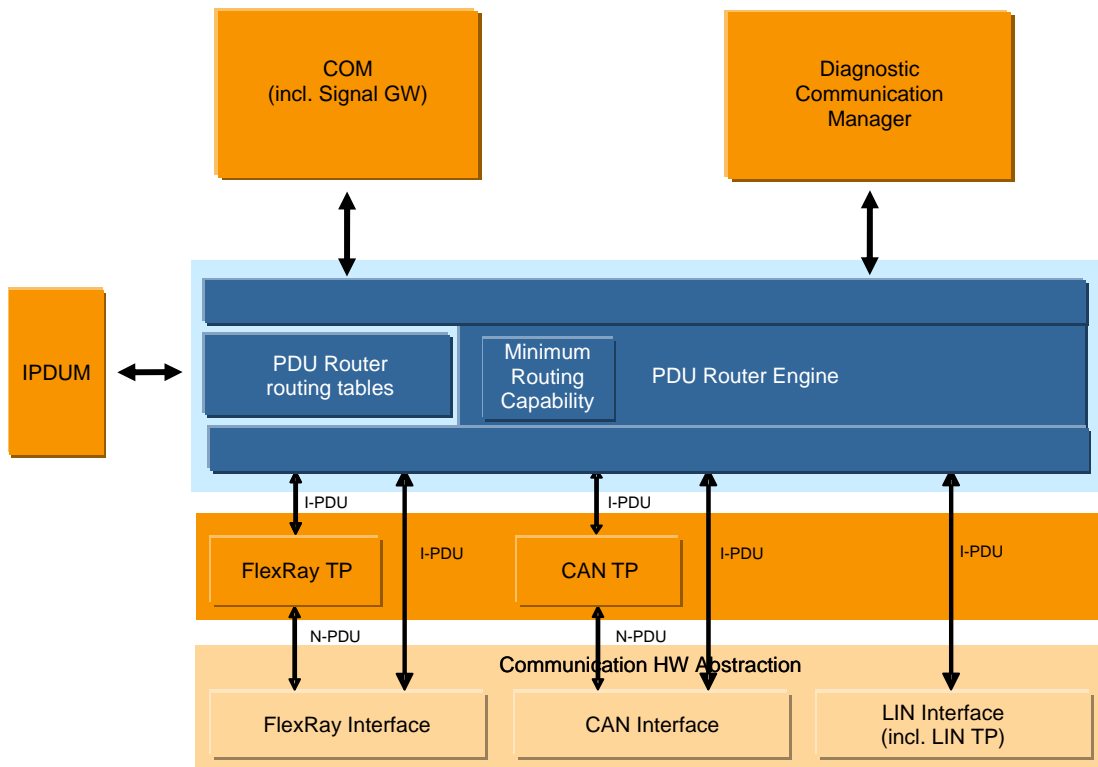
The detailed PDU Router module structure is shown in Figure 2. It mainly consists of two parts:

- The **PDU Router routing tables**: static routing tables describing the routing attributes for each PDU to be routed. The routing tables can be updated post-build time in the programming state of the ECU (see section 7.5).
- The **PDU Router Engine**: the actual code performing routing actions according to the PDU Router routing tables. The router engine has to deal with two translations:



- The **PDU Router UP Translation (PRUPT)**: Translation of PDU IDs and API of the PDU Router module to the related module above the PDU Router module (e.g.: COM, DCM, ...) or the IPDUM.
- The **PDU Router LO Translation (PRLT)**: Translation of PDU IDs and API of the PDU Router module to the related module below the PDU Router module (FlexRay Interface, CAN Interface, FlexRay TP, ...) or the IPDUM.

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



**Figure 2: Detailed PDU Router Structure**

## 2 Acronyms and abbreviations

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

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

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

### 3 Related documentation

#### 3.1 Input documents

- [1] Layered Software Architecture  
AUTOSAR\_LayeredSoftwareArchitecture.pdf,
- [2] Requirements on Gateway,  
AUTOSAR\_SRS\_Gateway.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_General.pdf
- [4] Specification of Standard Types  
AUTOSAR\_SWS\_StandardTypes.pdf
- [5] Specification of Communication Stack Types  
AUTOSAR\_SWS\_ComStackTypes.pdf
- [6] Specification of Development Error Tracer  
AUTOSAR\_SWS\_DevelopmentErrorTracer.pdf
- [7] Specification of CAN Interface  
AUTOSAR\_SWS\_CAN\_Interface.pdf
- [8] Specification of CAN Transport Layer  
AUTOSAR\_SWS\_CAN\_TP.pdf
- [9] Specification of LIN Interface  
AUTOSAR\_SWS\_LIN\_Interface.pdf
- [10] Specification of FlexRay Interface  
AUTOSAR\_SWS\_FlexRay\_Interface.pdf
- [11] Specification of FlexRay Transport Layer  
AUTOSAR\_SWS\_FlexRay\_TP.pdf
- [12] Specification of Communication  
AUTOSAR\_SWS\_COM.pdf
- [13] Specification of DCM  
AUTOSAR\_SWS\_DCM.pdf
- [14] Specification of DEM  
AUTOSAR\_SWS\_DEM.pdf
- [15] Specification of ECU Configuration  
AUTOSAR\_ECU\_Configuration.pdf

- [16] Specification of ECU ConfigurationParameters  
AUTOSAR\_ECU\_ConfigurationParameters.pdf
- [17] Specification of I-PDU Multiplexer  
AUTOSAR\_SWS\_IPDUM.pdf
- [18] AUTOSAR Basic Software Module Description Template,  
AUTOSAR\_BSWMDTemplate.pdf

### **3.2 Related standards and norms**

- [19] LIN Communication Protocol, LIN specification package, Revision 2.0,  
September 23, 2003
- [20] CAN Communication Protocol, ISO11898 – Road vehicles - Controller area  
network (CAN)
- [21] ISO 15765-2(2003-11-11), Road vehicles — Diagnostics on Controller Area  
Networks (CAN) — Part2: Network layer services
- [22] FlexRay Communication Protocol, FlexRay Communication Systems Protocol  
Specification Version 2.1

## 4 Constraints and assumptions

### 4.1 Limitations

1. The PDU Router module does not provide mechanisms for signal extraction or conversion.
2. The PDU Router module does not provide mechanisms for data integrity checking (like checksums).
3. The PDU Router module does not change or modify the I-PDU.
4. The PDU Router module does not make any PDU payload dependent routing decisions.
5. The PDU Router module does not support routing between TP modules and communication interface modules or vice versa.
6. The PDU Router module does not support 1:n routing of I-PDUs which are sent or received via a TP module and require multiple frames for transmission.
7. The PDU Router module itself does not support routing of I-PDUs between communication interface modules with rate conversion. (This functionality will be supported in cooperation with an upper layer module, e.g. COM as shown in section 9.4, Figure 15).

### 4.2 Applicability to car domains

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

## 5 Dependencies to other modules

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

- Communication interface modules:  
<Lo>If\_Transmit (e.g. CanIf\_Transmit, Frlf\_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:  
Ip dum\_Transmit  
Ip dum\_TxConfirmation  
Ip dum\_TriggerTransmit  
Ip dum\_RxIndication

### 5.1 File structure

#### 5.1.1 Code file structure

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

- PduR\_Cfg.c – for pre-compile time configuration parameters implemented as “const”,
- PduR\_Lcfg.c – for link time configurable parameters and
- PduR\_PBcfg.c – for post build time configurable parameters.

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

#### 5.1.2 Header file structure

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

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

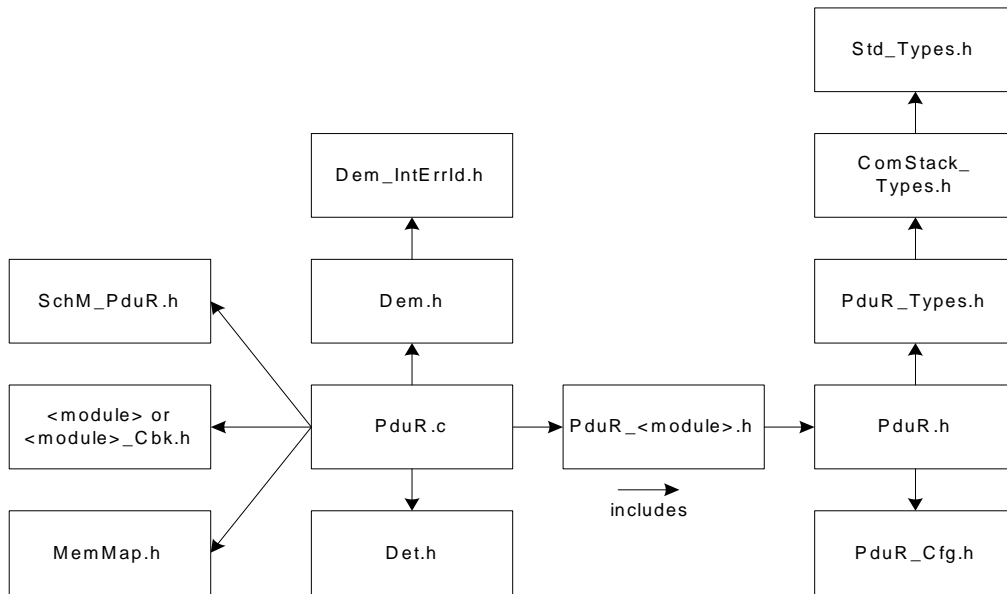
**PDUR159:** Pre-compile-time configuration data of the PDU Router module shall be defined in `PduR_Cfg.h`.

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

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

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

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



**Figure 3: File Structure**

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

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



## 6 Requirements traceability

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

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00323] API parameter checking	<a href="#">PDUR227</a> , <a href="#">PDUR221</a> , <a href="#">PDUR223</a> , <a href="#">PDUR224</a>
[BSW00336] Shutdown interface	not applicable
[BSW00337] Classification of errors	<a href="#">PDUR100</a> , <a href="#">PDUR101</a> , <a href="#">PDUR331</a> , <a href="#">PDUR332</a> , <a href="#">PDUR103</a>
[BSW00338] Detection and Reporting of development errors	<a href="#">PDUR100</a> , <a href="#">PDUR101</a> , <a href="#">PDUR331</a> , <a href="#">PDUR332</a> , <a href="#">PDUR104</a> , <a href="#">PDUR106</a> , <a href="#">PDUR221</a> , <a href="#">PDUR222</a> , <a href="#">PDUR223</a> , <a href="#">PDUR224</a> , <a href="#">PDUR231</a>
[BSW00339] Reporting of production relevant error status	<a href="#">PDUR103</a> , <a href="#">PDUR232</a> , <a href="#">PDUR233</a> , <a href="#">PDUR100</a> , <a href="#">PDUR255</a> , <a href="#">PDUR258</a>
[BSW00344] Reference to link-time configuration	<a href="#">PDUR240</a> , <a href="#">PDUR226</a>
[BSW00345] Pre-compile-time configuration	<a href="#">PDUR159</a> , <a href="#">PDUR226</a>
[BSW00369] Do not return development error codes via API	<a href="#">PDUR331</a> , <a href="#">PDUR332</a> , chapter 8
[BSW00375] Notification of wake-up reason	not applicable
[BSW00380] Separate C-File for configuration parameters	<a href="#">PDUR226</a>
[BSW00381] Separate configuration header file for pre-compile time parameters	<a href="#">PDUR159</a>
[BSW00383] List dependencies of configuration files	<a href="#">PDUR132</a>
[BSW00384] List dependencies to other modules	chapter 8.5
[BSW00385] List possible error notifications	<a href="#">PDUR100</a>
[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 <a href="#">PDUR242</a> , <a href="#">PDUR243</a> , <a href="#">PDUR245</a>
[BSW00398] Link-time parameters	chapter 10.2 <a href="#">PDUR242</a>
[BSW00399] Loadable Post-build time parameters	chapter 10.2 <a href="#">PDUR244</a> , <a href="#">PDUR246</a> , <a href="#">PDUR247</a> , <a href="#">PDUR248</a> , <a href="#">PDUR249</a>
[BSW004] Version check	Implementation requirement
[BSW00400] Selectable Post-build time parameters	not applicable
[BSW00402] Published information	<a href="#">PDUR236</a>
[BSW00404] Reference to post build time configuration	<a href="#">PDUR241</a> , <a href="#">PDUR226</a>
[BSW00405] Reference to multiple configuration sets	not applicable

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00406] Check module initialization	<a href="#">PDUR324</a> , <a href="#">PDUR325</a> , <a href="#">PDUR326</a> , <a href="#">PDUR327</a> , <a href="#">PDUR119</a>
[BSW00407] Function to read out published parameters	<a href="#">PDUR234</a>
[BSW00409] Header files for production code error IDs	<a href="#">PDUR132</a> , <a href="#">PDUR232</a>
[BSW00412] Separate H-File for configuration parameters	<a href="#">PDUR159</a>
[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	<a href="#">PDUR226</a>
[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	<a href="#">PDUR214</a> , 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	<a href="#">PDUR335</a> , <a href="#">PDUR336</a> , <a href="#">PDUR337</a>
[BSW159] Tool-based configuration	chapter 10
[BSW167] Static configuration checking	<a href="#">PDUR225</a>
[BSW168] Diagnostic Interface of SW components	not applicable
[BSW170] Data for reconfiguration of AUTOSAR SW-components	not applicable
[BSW171] Configurability of optional functionality	<a href="#">PDUR250</a> , <a href="#">PDUR242</a> , <a href="#">PDUR235</a> , chapter 10.2

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

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

[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	<a href="#">PDUR132</a>
---	-------------------------

Document: Requirements on Gateway

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

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

## 7 Functional Specification

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

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

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

### 7.1 General Behavior

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

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

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

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

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

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

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

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

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

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

**PDUR298:** The behaviour of all PDU Router module functions shall be synchronous although the overall behavior of a function might be asynchronous (e.g. a transmission request for CAN: PduR\_ComTransmit, Com\_TxConfirmation).

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

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

### 7.1.1 PDU Reception

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

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

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

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

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

### 7.1.2 PDU Transmission

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 7.1.3 PDU Gateway

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

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



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

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

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

It may happen that gatewayed I-PDUs will grow in size, e.g. when ECUs connected to the gatewayed are upgraded. If post-build is not used then it is costly to upgrade the gateway just to be able to process these frames. Therefore:

**PDUR491:** The PDU Router module shall copy the data of PDUs up to the received data length (ActualDLC).

**PDUR492:** In case of buffered routing, if the actualDLC is bigger than the configured buffer size, the PduRouter shall cut the message and only the part of the message that can be routed, shall be routed.

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

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

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

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

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

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

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

**PDUR310:** The PDU Router module shall maintain at least two values for each TT-FIFO: (1) the transmit confirmation pending flag (TxConfP), which indicates if a transmit confirmation is pending for the related I-PDU and (2) the index of the current FIFO entry (TxIdx), which indicates the FIFO entry which shall be used by the next PduR\_<Lo>IfTriggerTransmit call.

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

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

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

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

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

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

- (a) If TxConfP is not set, the PDU Router module shall call <Lo>If\_Transmit of the related interface module with the new I-PDU. If <Lo>If\_Transmit returns with success, the PDU Router module shall set TxConfP, otherwise it shall report the error PDUR\_E\_PDU\_INSTANCE\_LOST to DEM module.

- (b) If TxConfP is set and the FIFO is not full, the PDU Router module shall add the new I-PDU to the FIFO.
- (c) If TxConfP is set and the FIFO is full, the PDU Router module shall flush the FIFO (i.e. all entries shall be removed from the FIFO and TxConfP shall be cleared), report the error `PDUR_E_PDU_INSTANCE_LOST` to DEM module and process the new I-PDU according to rule (a).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 7.2 Zero Cost Operation

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

## 7.3 Minimum Routing

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

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

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

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

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

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

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

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

## 7.4 Reentrancy of API calls

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

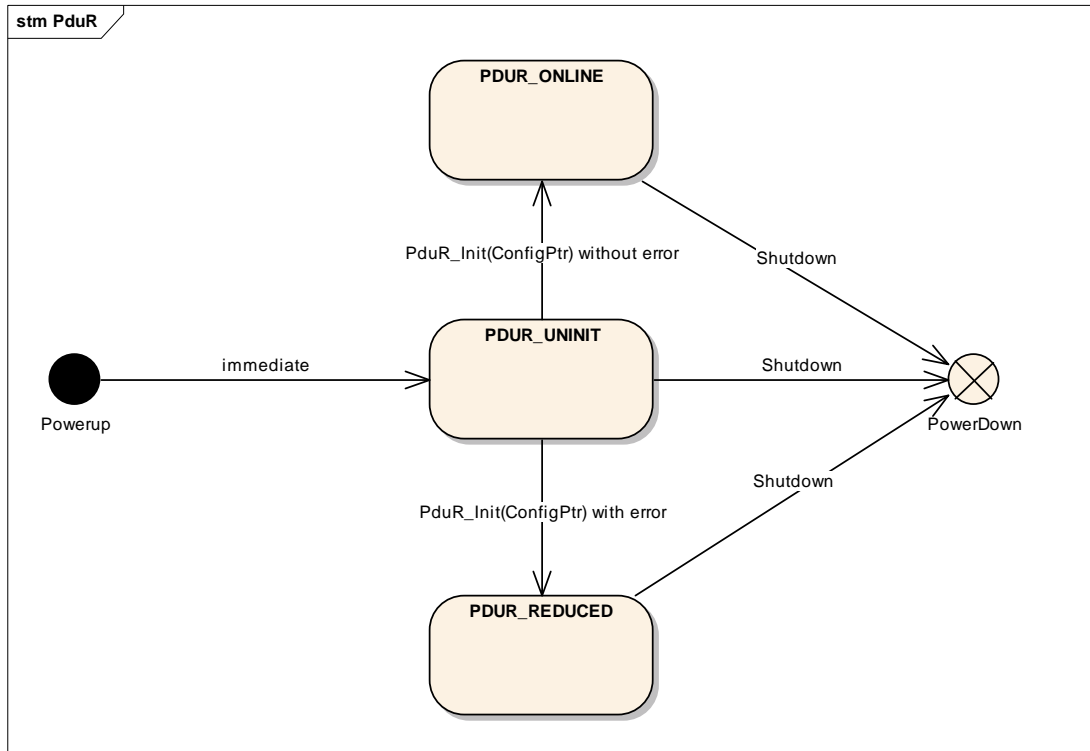
## 7.5 State Management

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

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

**PDUR326:** The PDU Router module shall change to the state PDUR\_ONLINE when the PDU Router has successfully been initialized via the function PduR\_Init.

**PDUR327:** The PDU Router module shall change to the state PDUR\_REDUCED in case the initialization within the function PduR\_Init did not succeed.



**Figure 4: PDU Router states**

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

**PDUR329:** The PDU Router module shall perform minimum routing when it is in the online state (PDUR\_ONLINE) or reduced state (PDUR\_REDUCED).

**PDUR330:** The PDU Router module shall perform no routing when it is in the uninitialised state (PDUR\_UNINIT).

### 7.6 Error classification

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

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

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

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

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

**PDUR231:** Development error values are of type uint8.

## 7.7 Error detection

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

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

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

**PDUR119:** If the PDU Router module has not been initialized (state PDUR\_UNINIT), all functions except `PduR_Init` shall report the error `PDUR_E_INVALID_REQUEST` via the Development Error Tracer (DET) when called.

## 7.8 Error notification

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

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

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

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

<code>ModuleId</code>	Module ID of the PDU Router: 51 decimal (see <a href="#">PDUR217</a> )
<code>InstanceId</code>	0 (single instance module)
<code>ApiId</code>	ID of API which reports an error: Service ID defined in section 8.3
<code>ErrorId</code>	ID of detected development error: value according to section 7.6

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

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

## 7.9 API parameter checking

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

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

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



## 8 API specification

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

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

### 8.1 Imported types

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

**PDUR333:**

<b>Header file</b>	<b>Imported Type</b>
BufReq_Types.h	BufReq_ReturnType
ComStack_Types.h	NotifResultType
	PdulType
	PduLengthType
Dem_Types.h	Dem_EventIdType
FrTp_Types.h	FrTp_NotifResultType
PrimitiveTypes.h	PduInfoType
Std_Types.h	Std_ReturnType
	Std_VersionInfoType

### 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

<b>Name:</b>	PduR_StateType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	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
<b>Description:</b>	States of the PDU Router	

**PDUR284:** The type `PduR_StateType` defines the PDU Router states.

#### 8.2.2 PduR\_LConfigType

<b>Name:</b>	PduR_LConfigType
<b>Type:</b>	Structure

<b>Range:</b>	Implementation specific.
<b>Description:</b>	Data structure containing link-time configuration data of the PDU Router

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

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

### 8.2.3 PduR\_PBConfigType

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

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

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

### 8.2.4 PduR\_CancelReasonType

**PDUR479:** PduR\_CancelReasonType

<b>Type:</b>	enum
<b>Range:</b>	PDU_CNLD0      Cancel Transfer because data are outdated
	PDU_CNLNB      Cancel Transfer because no further buffer can be provided
	PDU_CNLOR      Cancel Transfer because of another reason
<b>Description:</b>	The reason is sent to the other peer (not on receiver side in a 1:n connection) by the means of an appropriate FC frame.

### 8.2.5 PduR\_ParameterValueType

**PDUR480:** PduR\_ParameterValueType

<b>Type:</b>	uint8
<b>Range:</b>	0x00 – 0x7F; 0xF1 – PDUR_STmin 0xF9
<b>Description:</b>	Ranges of this parameter

## 8.3 Function definitions

### 8.3.1 General functions provided by the PDU Router

#### 8.3.1.1 PduR\_Init

##### PDUR334:

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

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

**PDUR336:** If the configuration parameter PduRZeroCostOperation is disabled and the PDU Router module is in the state PDUR\_UNINIT, the function PduR\_Init shall initialize the PDU Router module (e.g. PDU transmit buffers shall be initialized according to PDUR308, PDUR309, PDUR310, PDUR311 or PDUR312, PDUR313 depending on the PDU transmit buffer type).

**PDUR337:** If the configuration parameter PduRZeroCostOperation is disabled and the PDU Router module is not in the state PDUR\_UNINIT, the function PduR\_Init shall ignore the request and raise the error PDUR\_E\_INVALID\_REQUEST if development error detection is enabled.

**PDUR106:** The function PduR\_Init shall set the state of the PDU Router module to PDUR\_REDUCED and raise the error PDUR\_E\_INIT\_FAILED if the initialization of the PDU Router module failed.

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

**PDUR311:** The function PduR\_Init shall initialize the PDU transmit buffers which are configured as FIFO through parameter PduRFifoTxBufferSupport, Sets their related transmit confirmation pending flag (TxConfP) to Zero, copy the related configured

default value to the FIFO and set their FIFO entry (TxIdx) to this configured default value.

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

**PDUR222:** If development error detection is enabled, the function PduR\_Init shall raise development error PDUR\_E\_CONFIG\_PTR\_INVALID if ConfigPtr is NULL.

### 8.3.1.2 PduR\_GetVersionInfo

**PDUR338:**

<b>Service name:</b>	PduR_GetVersionInfo
<b>Syntax:</b>	void PduR_GetVersionInfo( Std_VersionInfoType* versionInfo )
<b>Service ID[hex]:</b>	0x17
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	versionInfo   Pointer to where to store the version information of this module.
<b>Return value:</b>	None
<b>Description:</b>	Returns the version information of this module.

**PDUR234:** The function PduR\_GetVersionInfo shall return the version information of this module. The version information includes:

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

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

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

**PDUR235:** The function PduR\_GetVersionInfo shall be pre-compile time configurable On/Off by the configuration parameter: PduRVersionInfoApi

### 8.3.1.3 PduR\_GetConfigurationId

**PDUR341:**

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

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

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

### 8.3.1.4 PduR\_CancelTransmitRequest

PDUR481: PduR\_CancelTransmitRequest

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

	corresponding channel.
<b>Description:</b>	This service primitive is used to cancel the transfer of pending I-PDUs. This function has to be called with the PDU-Id and the reason for cancellation.

### 8.3.1.5 PduR\_ChangeParameterRequest:

#### PDUR482: PduR\_ChangeParameterRequest

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

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

## 8.3.2 Function definitions for CAN interaction

### 8.3.2.1 PduR\_CanIfRxIndication

#### PDUR343:

<b>Service name:</b>	PduR_CanIfRxIndication	
<b>Syntax:</b>	<pre>void PduR_CanIfRxIndication(     PduIdType CanRxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x01	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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

	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Rx indicator for the CAN Interface	

The function PduR\_CanIfRxIndication is called by the CAN Interface after a CAN L-PDU has been received.

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

**PDUR344:** The function PduR\_CanIfRxIndication shall be callable in interrupt context.

**PDUR345:** The function PduR\_CanIfRxIndication shall be pre-compile time configurable On/Off by the configuration parameter: PduRCanIfSupport .

### 8.3.2.2 PduR\_CanIfTxConfirmation

#### PDUR346:

<b>Service name:</b>	PduR_CanIfTxConfirmation	
<b>Syntax:</b>	void PduR_CanIfTxConfirmation( PduIdType CanTxPduId )	
<b>Service ID[hex]:</b>	0x02	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Tx confirmation for the CAN Interface	

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

**PDUR194:** The function PduR\_CanIfTxConfirmation shall translate the CanTxPduId into the configured target PDU ID and route this confirmation to the configured target function. If the target PDU ID belongs to an interface module (gateway case) and is

statically configured to use a FIFO (PDU transmit buffer), the function PduR\_CanIfTxConfirmation shall process the confirmation according to PDUR259.

**PDUR347:** The function PduR\_CanIfTxConfirmation shall be callable in interrupt context (e.g. from CAN transmit interrupt). CanIf\_Transmit() may be called within PduR\_CanIfTxConfirmation.

**PDUR348:** The function PduR\_CanIfTxConfirmation must not be called in the context of CanIf\_Transmit.

**PDUR349:** The function PduR\_CanIfTxConfirmation shall be pre-compile time configurable *On/Off* by the configuration parameter: PduRCanIfSupport .

### 8.3.2.3 PduR\_CanTpProvideRxBuffer

#### PDUR350:

<b>Service name:</b>	PduR_CanTpProvideRxBuffer	
<b>Syntax:</b>	<pre>BufReq_ReturnType PduR_CanTpProvideRxBuffer(     PduIdType CanTpRxPduId,     PduLengthType TpSduLength,     PduInfoType** PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x03	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same Pduld.	
<b>Parameters (in):</b>	CanTpRxPdulId	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 CanTpRxPdulId. The length will be greater than zero.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	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.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer available BUFREQ_E_OVFL: Receiver is not able to receive number of TpSduLength bytes; no buffer provided. BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided..
<b>Description:</b>	Provides Rx buffer for the CAN TP.	

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



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

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

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

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

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

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

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

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

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

### 8.3.2.4 PduR\_CanTpRxIndication

**PDUR353:**

<b>Service name:</b>	<code>PduR_CanTpRxIndication</code>
<b>Syntax:</b>	<pre>void PduR_CanTpRxIndication(     PduIdType CanTpRxPduId,     NotifResultType Result )</pre>
<b>Service ID[hex]:</b>	<code>0x04</code>
<b>Sync/Async:</b>	Synchronous

<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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.  <ul style="list-style-type: none"> <li>• NTFRSLT_OK , NTFRSLT_E_CANCELLATION_OK in case TP reception completed successfully</li> <li>• NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Cr, NTFRSLT_E_WRONG_SN, NTFRSLT_E_UNEXP_PDU, NTFRSLT_E_NO_BUFFER in case TP reception did not complete successfully (e.g. because of a timeout); used to enable unlocking of the receive buffer</li> </ul>
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Rx indicator for the CAN TP	

The function PduR\_CanTpRxIndication is called by the CAN TP.

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

**PDUR441:** The function PduR\_CanTpRxIndication shall translate the CanTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

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

**PDUR354:** The function PduR\_CanTpRxIndication shall be callable in interrupt context.

**PDUR355:** The function PduR\_CanTpRxIndication shall be pre-compile time configurable On/Off by the configuration parameter: PduRCanTpSupport .

### 8.3.2.5 PduR\_CanTpProvideTxBuffer

**PDUR356:**

<b>Service name:</b>	PduR_CanTpProvideTxBuffer
----------------------	---------------------------

<b>Syntax:</b>	BufReq_ReturnType PduR_CanTpProvideTxBuffer( PduIdType CanTpTxPduId, PduInfoType** PduInfoPtr, uint16 Length )	
<b>Service ID[hex]:</b>	0x05	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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).
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	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.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer of the requested size is available BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
<b>Description:</b>	Provides Tx buffer for the CAN TP.	

The function PduR\_CanTpProvideTxBuffer is called by the CAN TP for requesting a transmit buffer.

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

**PDUR443:** The function PduR\_CanTpProvideTxBuffer shall translate the CanTpTxPduId into the configured target PDU ID and route this request to the configured target function.

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

**PDUR357:** The function PduR\_CanTpProvideTxBuffer shall be callable in interrupt context.

**PDUR358:** The function PduR\_CanTpProvideTxBuffer shall be pre-compile time configurable On/Off by the configuration parameter: PduRCanTpSupport.

Caveats of PduR\_CanTpProvideTxBuffer: In case the function PduR\_CanTpProvideTxBuffer returns BUFREQ\_E\_NOT\_OK the related transmit request is not finished. The related TP module may either finish the request by providing a final confirmation indicating an error or may retry the buffer request.

### 8.3.2.6 PduR\_CanTpTxConfirmation

#### PDUR359:

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

The function PduR\_CanTpTxConfirmation is called by the CAN Transport Protocol:

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

**PDUR445:** The function PduR\_CanTpTxConfirmation shall translate the CanTpTxPduId into the configured target PDU ID and route this indication to the configured target function.

**PDUR446:** If CanTpTxPduId belongs to a gateway operation the function PduR\_CanTpTxConfirmation shall use this indication to unlock the transmit buffer.

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

**PDUR360:** The function PduR\_CanTpTxConfirmation shall be callable in interrupt context (e.g. from CAN transmit interrupt).

**PDUR361:** The function PduR\_CanTpTxConfirmation shall be pre-compile time configurable On/Off by the configuration parameter: PduRCanTpSupport .

### 8.3.2.7 PduR\_CanTpChangeParameterConfirmation

**PDUR483:**

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

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

### 8.3.3 Function definitions for FlexRay interaction

#### 8.3.3.1 PduR\_FrlfRxIndication

##### PDUR362:

<b>Service name:</b>	PduR_FrlfRxIndication	
<b>Syntax:</b>	<pre>void PduR_FrlfRxIndication(     PduIdType FrRxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x07	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Rx indicator for the FlexRay Interface	

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

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

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

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

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

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

### 8.3.3.2 PduR\_FrlfTxConfirmation

#### PDUR365:

<b>Service name:</b>	PduR_FrlfTxConfirmation	
<b>Syntax:</b>	void PduR_FrlfTxConfirmation( PduIdType FrTxPduId )	
<b>Service ID[hex]:</b>	0x08	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same Pduld.	
<b>Parameters (in):</b>	FrTxPdulId	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
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Tx confirmation for the FlexRay Interface	

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

**PDUR450:** The function PduR\_FrlfTxConfirmation shall translate the FrTxPdulId into the configured target PDU ID and route this confirmation to the configured target function.

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

**PDUR366:** The function PduR\_FrlfTxConfirmation shall be callable in interrupt context (e.g. from the FlexRay transmit interrupt).

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

**PDUR367:** The function PduR\_FrlfTxConfirmation must not be called in the context of Frlf\_Transmit. Frlf\_Transmit() may be called within PduR\_FrlfTxConfirmation.

**PDUR368:** The function PduR\_FrlfTxConfirmation shall be pre-compile time configurable On/Off by the configuration parameter: PduRFrIfSupport.

### 8.3.3.3 PduR\_FrlfTriggerTransmit

#### PDUR369:

<b>Service name:</b>	PduR_FrlfTriggerTransmit	
<b>Syntax:</b>	Std_ReturnType PduR_FrlfTriggerTransmit( PduIdType FrTxPduId, PduInfoType* PduInfoPtr )	
<b>Service ID[hex]:</b>	0x09	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
<b>Parameters (inout):</b>	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
<b>Description:</b>	Triggers the transmission of a FlexRay frame	

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

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

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

**PDUR370:** The function PduR\_FrlfTriggerTransmit shall be callable in interrupt context.

**PDUR371:** The function PduR\_FrlfTriggerTransmit shall be pre-compile time configurable On/Off by the configuration parameter: PduRFrIfSupport .

### 8.3.3.4 PduR\_FrTpProvideRxBuffer



**PDUR372:**

<b>Service name:</b>	PduR_FrTpProvideRxBuffer	
<b>Syntax:</b>	BufReq_ReturnType PduR_FrTpProvideRxBuffer( PduIdType FrTpRxPduId, PduLengthType TpSduLength, PduInfoType** PduInfoPtr )	
<b>Service ID[hex]:</b>	0x0a	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same Pduld.	
<b>Parameters (in):</b>	FrTpRxPdulId	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 FrTpRxPdulId The length will be greater than zero.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	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.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer available BUFREQ_E_OVFL: Receiver is not able to receive number of TpSduLength bytes; no buffer provided. BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
<b>Description:</b>	Provides Rx Buffer for the FlexRay TP	

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

**PDUR454:** The function PduR\_FrTpProvideRxBuffer shall translate the FrTpRxPdulId into the configured target PDU ID and route this request to the configured target function.

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

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

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

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

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

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

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

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

### 8.3.3.5 PduR\_FrTpRxIndication

**PDUR375:**

<b>Service name:</b>	<code>PduR_FrTpRxIndication</code>	
<b>Syntax:</b>	<pre>void PduR_FrTpRxIndication(     PduIdType FrTpRxPduId,     NotifResultType Result )</pre>	
<b>Service ID[hex]:</b>	0x0b	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	<code>FrTpRxPduId</code>	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. <ul style="list-style-type: none"> <li>• <code>NTFRSLT_OK</code>, <code>NTFRSLT_E_CANCELLATION_OK</code> in case TP reception completed successfully,</li> <li>• <code>NTFRSLT_E_NOT_OK</code>, <code>NTFRSLT_E_CANCELLATION_NOT_OK</code>, <code>NTFRSLT_E_TIMEOUT_A</code>, <code>NTFRSLT_E_TIMEOUT_Cr</code>, <code>NTFRSLT_E_WRONG_SN</code>, <code>NTFRSLT_E_UNEXP_PDU</code>, <code>NTFRSLT_E_NO_BUFFER</code> in case TP reception did not complete successfully (e.g. because of a timeout); used to enable unlocking of the receive buffer</li> </ul>
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	

<b>Description:</b>	Rx indicator for the FlexRay TP
---------------------	---------------------------------

The function PduR\_FrTpRxIndication is called by the FlexRay

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

**PDUR456:** The function PduR\_FrTpRxIndication shall translate the FrTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

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

**PDUR376:** The function PduR\_FrTpRxIndication shall be callable in interrupt context.

**PDUR377:** The function PduR\_FrTpRxIndication shall be pre-compile time configurable On/Off by the configuration parameter: PduRFrTpSupport .

### 8.3.3.6 PduR\_FrTpProvideTxBuffer

**PDUR378:**

<b>Service name:</b>	PduR_FrTpProvideTxBuffer	
<b>Syntax:</b>	<pre>BufReq_ReturnType PduR_FrTpProvideTxBuffer(     PduIdType FrTpTxPduId,     PduInfoType** PduInfoPtr,     uint16 Length )</pre>	
<b>Service ID[hex]:</b>	0x0c	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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).
<b>Parameters (inout):</b>	None	

<b>Parameters (out):</b>	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.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer of the requested size is available BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
<b>Description:</b>	ProvidesTx Buffer for the FlexRay TP	

The function PduR\_FrTpProvideTxBuffer is called by the FlexRay TP for requesting a transmit buffer.

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

**PDUR458:** The function PduR\_FrTpProvideTxBuffer shall translate the FrTpTxPduId into the configured target PDU ID and route this request to the configured target function.

**PDUR459:** If FrTpTxPduId belongs to a gateway operation the function PduR\_FrTpProvideTxBuffer itself has to provide the requested buffer. Therefor the function PduR\_FrTpProvideTxBuffer shall use the receive buffer which has previously been filled by the receiving TP module.

**PDUR379:** The function PduR\_FrTpProvideTxBuffer shall be callable in interrupt context.

**PDUR380:** The function PduR\_FrTpProvideTxBuffer shall be pre-compile time configurable On/Off by the configuration parameter: PduRFrTpSupport.

Caveats of PduR\_FrTpProvideTxBuffer: In case the function PduR\_FrTpProvideTxBuffer returns BUFREQ\_E\_NOT\_OK the related transmit request is not finished. The related TP module may either finish the request by providing a final confirmation indicating an error or may retry the buffer request.

### 8.3.3.7 PduR\_FrTpTxConfirmation

**PDUR381:**

<b>Service name:</b>	PduR_FrTpTxConfirmation
<b>Syntax:</b>	void PduR_FrTpTxConfirmation( PduIdType FrTpTxPduId, NotifResultType Result )
<b>Service ID[hex]:</b>	0x0d

<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	FrTpTxPduId	ID of FlexRay N-PDU that has been transmitted.  Range: 0..(maximum number of N-PDU IDs which may be transmitted by FlexRay TP) - 1
	Result	Result of the TP transmission: <ul style="list-style-type: none"> <li>• NTFRSLT_OK, NTFRSLT_E_CANCELLATION_OK in case TP transmission completed successfully,</li> <li>• NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Bs, NTFRSLT_E_INVALID_FS, NTFRSLT_E_WFT_OVRN, NTFRSLT_E_NO_BUFFER in case TP transmission did not complete successfully (e.g. because of a timeout); used to enable unlocking of the transmit buffer.</li> </ul>
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Tx confirmation for the FlexRay TP	

The function PduR\_FrTpTxConfirmation is called by the FlexRay TP:

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

**PDUR460:** The function PduR\_FrTpTxConfirmation shall translate the FrTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

**PDUR461:** If FrTpRxPduId belongs to a gateway operation the function PduR\_FrTpTxConfirmation shall use this indication to unlock the transmit buffer.

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

**PDUR382:** The function PduR\_FrTpTxConfirmation shall be callable in interrupt context (e.g. from FlexRay transmit interrupt).

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

**PDUR383:** The function PduR\_FrTpTxConfirmation shall be pre-compile time configurable On/Off by the configuration parameter: PduRFrTpSupport .

### 8.3.3.8 PduR\_FrTpChangeParameterConfirmation

PDUR484:

<b>Service name:</b>	PduR_FrTpChangeParameterConfirmation	
<b>Syntax:</b>	<pre>void PduR_FrTpChangeParameterConfirmation(     PduIdType FrTpTxPduId,     NotifResultType Result )</pre>	
<b>Service ID[hex]:</b>	0x1f	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	FrTpTxPduId	ID of FlexRay N-PDU that parameter has been changed. Range: 0..(maximum number of N-PDU IDs which may be transmitted by FlexRay TP) - 1
	Result	Result of the change parameter: <ul style="list-style-type: none"> <li>• NTFRSLT_PARAMETER_OK in case parameter changed successfully,</li> <li>• NTFRSLT_E_PARAMETER_NOT_OK, in case Parameter change</li> </ul>

	did not complete successfully.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Change Parameter confirmation for the FlexRay TP

The function PduR\_FrTpChangeParameterConfirmation shall be pre-compile time configurable On/Off by the configuration parameter:

PDUR\_FRTPCCHANGEPARAMETER\_SUPPORT.

### 8.3.4 Function definitions for LIN interaction

#### 8.3.4.1 PduR\_LinIfRxIndication

PDUR384:

<b>Service name:</b>	PduR_LinIfRxIndication	
<b>Syntax:</b>	<pre>void PduR_LinIfRxIndication(     PduIdType LinRxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x0e	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Rx indicator for the LIN Interface	

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

**PDUR463:** The function PduR\_LinIfRxIndication shall translate the LinRxPduId into the configured target PDU ID and route this indication to the configured target function.

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

**PDUR385:** The function PduR\_LinIfRxIndication shall be callable in interrupt context.

**PDUR386:** The function PduR\_LinIfRxIndication shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinIfSupport .

### 8.3.4.2 PduR\_LinIfTxConfirmation

**PDUR387:**

<b>Service name:</b>	PduR_LinIfTxConfirmation
<b>Syntax:</b>	void PduR_LinIfTxConfirmation( PduIdType LinTxPduId )
<b>Service ID[hex]:</b>	0x0f
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.
<b>Parameters (in):</b>	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
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Tx confirmation for the LIN Interface

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

**PDUR465:** The function PduR\_LinIfTxConfirmation shall translate the LinTxPduId into the configured target PDU ID and route this confirmation to the configured target function.

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



**PDUR388:** The function PduR\_LinIfTxConfirmation shall be callable in interrupt context.

**PDUR389:** The function PduR\_LinIfTxConfirmation must not be called in the context of LinIf\_Transmit. LinIf\_Transmit() may be called within PduR\_LinIfTxConfirmation.

**PDUR390:** The function PduR\_LinIfTxConfirmation shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinIfSupport.

### 8.3.4.3 PduR\_LinIfTriggerTransmit

**PDUR391:**

<b>Service name:</b>	PduR_LinIfTriggerTransmit	
<b>Syntax:</b>	Std_ReturnType PduR_LinIfTriggerTransmit( PduIdType LinTxPduId, PduInfoType* PduInfoPtr )	
<b>Service ID[hex]:</b>	0x10	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
<b>Parameters (inout):</b>	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
<b>Description:</b>	Triggers the transmission of a LIN frame	

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

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

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

**PDUR392:** The function PduR\_LinIfTriggerTransmit shall be callable in interrupt context.

**PDUR393:** The function PduR\_LinIfTriggerTransmit shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinIfSupport .

### 8.3.4.4 PduR\_LinTpProvideRxBuffer

**PDUR394:**

<b>Service name:</b>	PduR_LinTpProvideRxBuffer	
<b>Syntax:</b>	<pre>BufReq_ReturnType PduR_LinTpProvideRxBuffer(     PduIdType LinTpRxPduId,     PduLengthType TpSduLength,     PduInfoType** PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x11	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same Pduld.	
<b>Parameters (in):</b>	LinTpRxPdul	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 LinTpRxPdul. The length will be greater than zero.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	PduInfoPtr	Pointer to pointer to PdulInfoStructure 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.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer available BUFREQ_E_OVFL: Receiver is not able to receive number of TpSduLength bytes; no buffer provided. BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
<b>Description:</b>	Provides Rx Buffer for the LIN TP	

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

**PDUR469:** The function PduR\_LinTpProvideRxBuffer shall translate the LinTpRxPdul into the configured target PDU ID and route this request to the configured target function.

**PDUR470:** If the target PDU ID belongs to a TP module (gateway case) or there is more than one receiver (multicast case), the function PduR\_LinTpProvideRxBuffer

itself has to provide the requested buffer and shall forward the data of the receive buffer to the related receiver(s) when the buffer has been released by a further call of this function.

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

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

**PDUR395:** The function PduR\_LinTpProvideRxBuffer shall be callable in interrupt context.

**PDUR396:** The function PduR\_LinTpProvideRxBuffer shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinTpSupport.

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

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

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

### 8.3.4.5 PduR\_LinTpRxIndication

**PDUR397:**

<b>Service name:</b>	PduR_LinTpRxIndication	
<b>Syntax:</b>	<pre>void PduR_LinTpRxIndication(     PduIdType LinTpRxPduId,     NotifResultType Result )</pre>	
<b>Service ID[hex]:</b>	0x12	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	LinTpRxPduId	ID of LIN N-PDU that has been received. Range: 0..(maximum number of N-PDU IDs which may be received by LIN TP) - 1
	Result	Result of the TP reception. • NTFRSLT_OK , NTFRSLT_E_CANCELLATION_OK in case TP

	reception completed successfully; • NTFRSLT_E_NOT_OK, NTFRSLT_E_CANCELLATION_NOT_OK, NTFRSLT_E_TIMEOUT_A, NTFRSLT_E_TIMEOUT_Cr, NTFRSLT_E_WRONG_SN, NTFRSLT_E_UNEXP_PDU, NTFRSLT_E_NO_BUFFER in case TP reception did not complete successfully (e.g. because of a timeout); used to enable unlocking of the receive buffer
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Rx indicator for the LIN TP

The function PduR\_LinTpRxIndication is called by the LIN TP

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

**PDUR471:** The function PduR\_LinTpRxIndication shall translate the LinTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

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

**PDUR398:** The function PduR\_LinTpRxIndication shall be callable in interrupt context.

**PDUR399:** The function PduR\_LinTpRxIndication shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinTpSupport.

### 8.3.4.6 PduR\_LinTpProvideTxBuffer

**PDUR400:**

<b>Service name:</b>	PduR_LinTpProvideTxBuffer
<b>Syntax:</b>	BufReq_ReturnType PduR_LinTpProvideTxBuffer( PduIdType LinTpTxPduId, PduInfoType** PduInfoPtr, uint16 Length )
<b>Service ID[hex]:</b>	0x13
<b>Sync/Async:</b>	Synchronous

<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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).
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	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.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful BUFREQ_E_BUSY: Currently no buffer of the requested size is available BUFREQ_E_NOT_OK: Buffer request not successful, no buffer provided.
<b>Description:</b>	Provides Tx Buffer for the LIN TP	

The function PduR\_LinTpProvideTxBuffer is called by the LIN TP for requesting a transmit buffer.

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

**PDUR473:** The function PduR\_LinTpProvideTxBuffer shall translate the LinTpTxPduId into the configured target PDU ID and route this request to the configured target function.

**PDUR474:** If LinTpTxPduId belongs to a gateway operation the function PduR\_LinTpProvideTxBuffer itself has to provide the requested buffer. Therefore the function PduR\_LinTpProvideTxBuffer shall use the receive buffer which has previously been filled by the receiving TP module.

**PDUR401:** The function PduR\_LinTpProvideTxBuffer shall be callable in interrupt context.

In case function PduR\_LinTpProvideTxBuffer returns BUFREQ\_E\_NOT\_OK the related transmit request is not finished. The related TP module may either finish the request by providing a final confirmation indicating an error or may retry the buffer request.

**PDUR402:** The function PduR\_LinTpProvideTxBuffer shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinTpSupport.

### 8.3.4.7 PduR\_LinTpTxConfirmation

**PDUR403:**

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

The function PduR\_LinTpTxConfirmation is called by the LIN TP:

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

**PDUR475:** The function PduR\_LinTpTxConfirmation shall translate the LinTpRxPduId into the configured target PDU ID and route this indication to the configured target function.

**PDUR476:** If LinTpRxPduId belongs to a gateway operation the function PduR\_LinTpTxConfirmation shall use this indication to unlock the transmit buffer.

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

**PDUR404:** The function PduR\_LinTpTxConfirmation shall be callable in interrupt context.

**PDUR405:** The function PduR\_LinTpTxConfirmation shall be pre-compile time configurable On/Off by the configuration parameter: PduRLinTpSupport.

### 8.3.4.8 PduR\_LinTpChangeParameterConfirmation

PDUR485:

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

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



### 8.3.5 Function definitions for COM interaction

#### 8.3.5.1 PduR\_ComTransmit

##### PDUR406:

<b>Service name:</b>	PduR_ComTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType PduR_ComTransmit(     PduIdType ComTxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x15	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Transmit request has been accepted E_NOT_OK: Transmit request has not been accepted
<b>Description:</b>	Requests a transmission for the AUTOSAR COM Module	

The function PduR\_ComTransmit is called by AUTOSAR COM to request a transmission.

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

**PDUR218:** If ComTxPduId represents a group of PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error, the function PduR\_ComTransmit shall return E\_NOT\_OK.

**PDUR407:** The function PduR\_ComTransmit shall be pre-compile time configurable On/Off by the configuration parameter: PduRComSupport .

### 8.3.6 Function definitions for DCM interaction

#### 8.3.6.1 PduR\_DcmTransmit

##### PDUR408:

<b>Service name:</b>	PduR_DcmTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType PduR_DcmTransmit(     PduIdType DcmTxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x16	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Transmit request has been accepted E_NOT_OK: Transmit request has not been accepted
<b>Description:</b>	Requests a transmission for the DCM module.	

The function PduR\_DcmTransmit is called by the DCM to request a transmission.

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

**PDUR410:** The function PduR\_DcmTransmit shall only use the SduLength information within the parameter PduInfoPtr.

**PDUR411:** The function PduR\_DcmTransmit must not use the parameter PduInfoPtr because it is a pointer to undefined data

For a TP transmission request the function call PduR\_DcmTransmit will be followed by at least one invocation of PduR\_<Lo>TpProvideTxBuffer by the TP module to get the data (transmission buffer).

The reason for having the PduInfoPtr is to reach compliance with the COM API PduR\_ComTransmit().

**PDUR206:** If the parameter DcmTxPduId of the function PduR\_DcmTransmit represents a group of single frame TP PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error, the function PduR\_DcmTransmit shall return E\_NOT\_OK.

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

### 8.3.7 Function definitions for IPDUM interaction

#### 8.3.7.1 PduR\_IpdumTransmit

##### PDUR413:

<b>Service name:</b>	PduR_IpdumTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType PduR_IpdumTransmit(     PduIdType IpdumTxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x19	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	IpdumTxPduId	ID of IPDUM I-PDU to be transmitted. Range: 0..(maximum number of I-PDU IDs which may be transmitted by IPDUM) – 1
	PduInfoPtr	A pointer to a structure with I-PDU related data that shall be transmitted: data length and pointer to I-SDU buffer
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Transmit request has been accepted E_NOT_OK: Transmit request has not been accepted
<b>Description:</b>	Requests a transmission for the IPDUM	

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

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

**PDUR238:** If the parameter IpdumUpTxPduId of the function PduR\_IpdumTransmit represents a group of PDUs (multicast transmit request) and at least one of the forwarded transmit requests returns with an error, the function PduR\_IpdumTransmit shall return E\_NOT\_OK.

**PDUR414:** The function PduR\_IpdumTransmit shall be pre-compile time configurable On/Off by the configuration parameter: PduRIPduMSupport .

#### 8.3.7.2 PduR\_IpdumTxConfirmation

##### PDUR415:

<b>Service name:</b>	PduR_IpdumTxConfirmation	
<b>Syntax:</b>	<pre>void PduR_IpdumTxConfirmation(     PduIdType IpdumLoTxPduId</pre>	

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

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

**PDUR416:** The function PduR\_IpduMtxConfirmation shall translate the IpduMLoTxPduId into the configured target PDU ID and route this confirmation to the configured upper layer module (e.g. COM).

**PDUR417:** The function PduR\_IpduMtxConfirmation shall be pre-compile time configurable On/Off by the configuration parameter: PduRIPduMSupport.

**PDUR418:** The function PduR\_IpduMtxConfirmation shall be callable in interrupt context.

### 8.3.7.3 PduR\_IpduMrxIndication

**PDUR419:**

<b>Service name:</b>	PduR_IpduMrxIndication	
<b>Syntax:</b>	<pre>void PduR_IpduMrxIndication(     PduIdType IpduMLoRxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x1b	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	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
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Rx indicator for the IPDUM	

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

**PDUR420:** The function PduR\_IpdumRxIndication shall translate the parameter IpdumLoRxPduld into the configured target PDU ID and route this indication to the configured upper layer module (e.g. COM).

**PDUR421:** The function PduR\_IpdumRxIndication shall be callable in interrupt context.

**PDUR422:** The function PduR\_IpdumRxIndication shall be pre-compile time configurable On/Off by the configuration parameter: PduRIPduMSupport.

## 8.4 Scheduled functions

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

## 8.5 Expected Interfaces

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

### 8.5.1 Mandatory Interfaces

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

#### PDUR423:

<i>API function</i>	<i>Description</i>
Dem_ReportErrorStatus	Reports errors to the DEM.

### 8.5.2 Optional Interfaces

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

#### PDUR424:

<i>API function</i>	<i>Description</i>
---------------------	--------------------

IpDum_TxConfirmation	This function is called by the lower layer after the I-PDU has been transmitted on the network.
CanTp_Transmit	This service is used to request the transfer of segmented data.
Det_ReportError	Service to report development errors.
LinTp_Transmit	Requests the transfer of segmented data.
FrIf_Transmit	Requests the sending of a PDU.
FrTp_Transmit	<p>This service is utilized to request the transfer of data. It sets a flag for indicating that a transmit request is present.</p> <p>This function has to be called with the PDU-Id of the FrTp, i.e. the upper layer has to translate its own PDU-Id into the one of the TP for the corresponding message.</p> <p>Within the provided PduInfoPtr only SduLength is valid (no data)! If this function returns E_OK then there will arise an call of PduR_FrTpProvideTxBuffer in order to get data for sending.</p>
IpDum_Transmit	Service is called by the PDU-Router to request a transmission.
Com_TxConfirmation	<p>This function is called by the lower layer after the PDU has been transmitted on the network.</p> <p>A confirmation that is received for an I-PDU that does not require a confirmation is silently discarded.</p>
Com_TriggerTransmit	This function is called by the lower layer when an AUTOSAR COM I-PDU shall be transmitted. Within this function, AUTOSAR COM shall copy the contents of its I-PDU transmit buffer to the L-PDU buffer given by SduDataPtr.
Com_RxIndication	This function is called by the lower layer after an I-PDU has been received.
IpDum_RxIndication	Service is called when a multiplexed SDU has to be de-multiplexed.
IpDum_TriggerTransmit	Service is called by the lower layer when an IPduM I-PDU shall be transmitted.
CanIf_Transmit	--
Dcm_TxConfirmation	This is called by the PduR to confirm a Transmit
Dcm_ProvideRxBuffer	By this service the DCM module is requested to provide a buffer for a transport layer on first frame or single frame reception.
Dcm_RxIndication	This is called by the PduR to indicate the completion of a reception
Dcm_ProvideTxBuffer	By this service the DCM module is requested to provide a buffer containing data to be transmitted via a transport protocol.

### 8.5.3 Configurable interfaces

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

## 9 Sequence diagrams

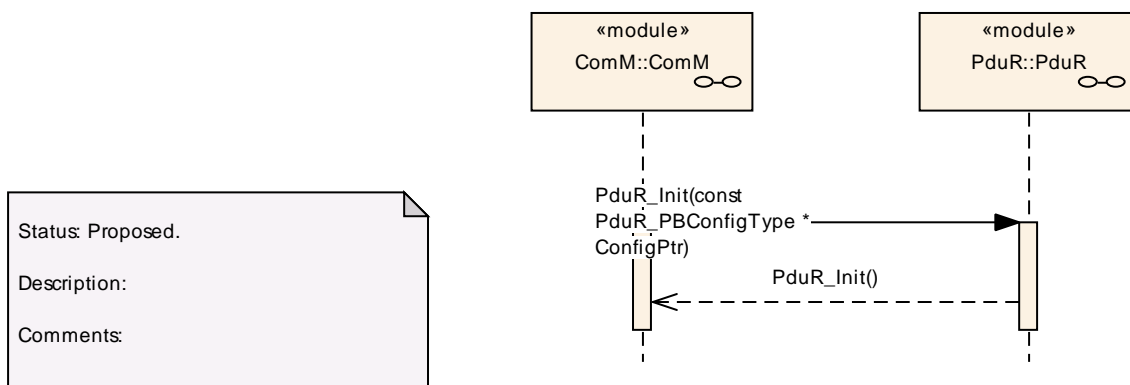
The goal of this chapter is to make the understanding of the PDU Router easier. For this purpose sequence diagrams which show different communication scenarios are used. Please consider that the sequence diagrams are not exhaustive and are only used to support the functional specification (chapter 7)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

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

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

### 9.1 Initialization

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

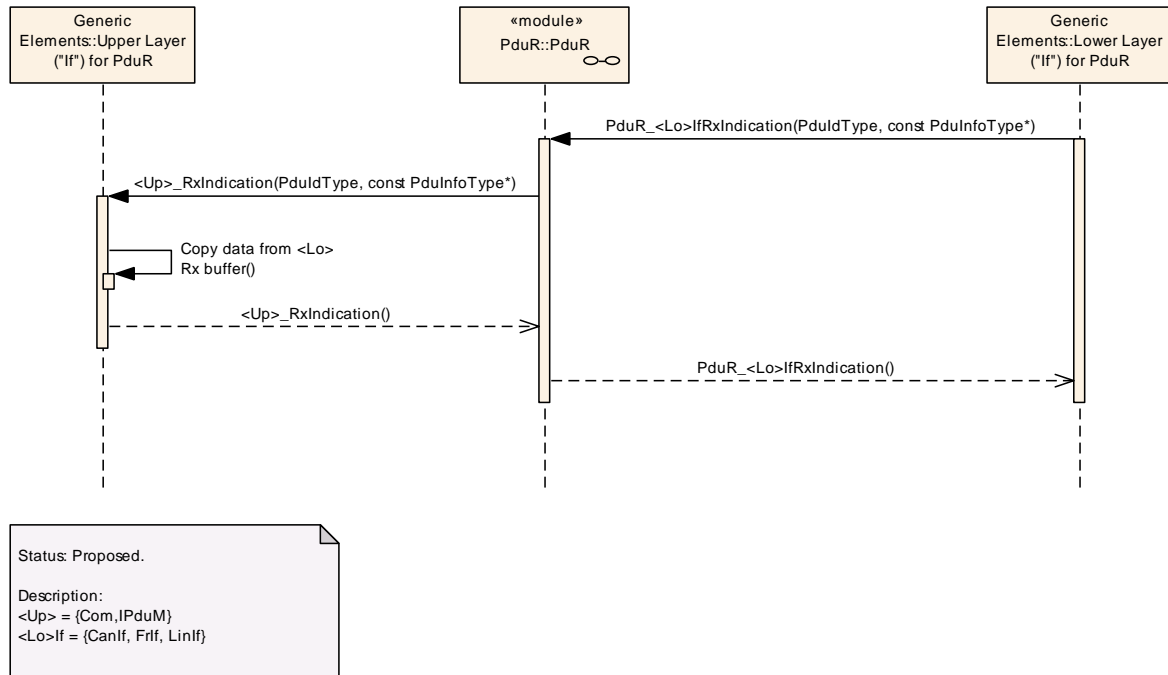


**Figure 5: Initialization**



## 9.2 PDU Reception

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



**Figure 6: non TP-PDU-Rx**

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

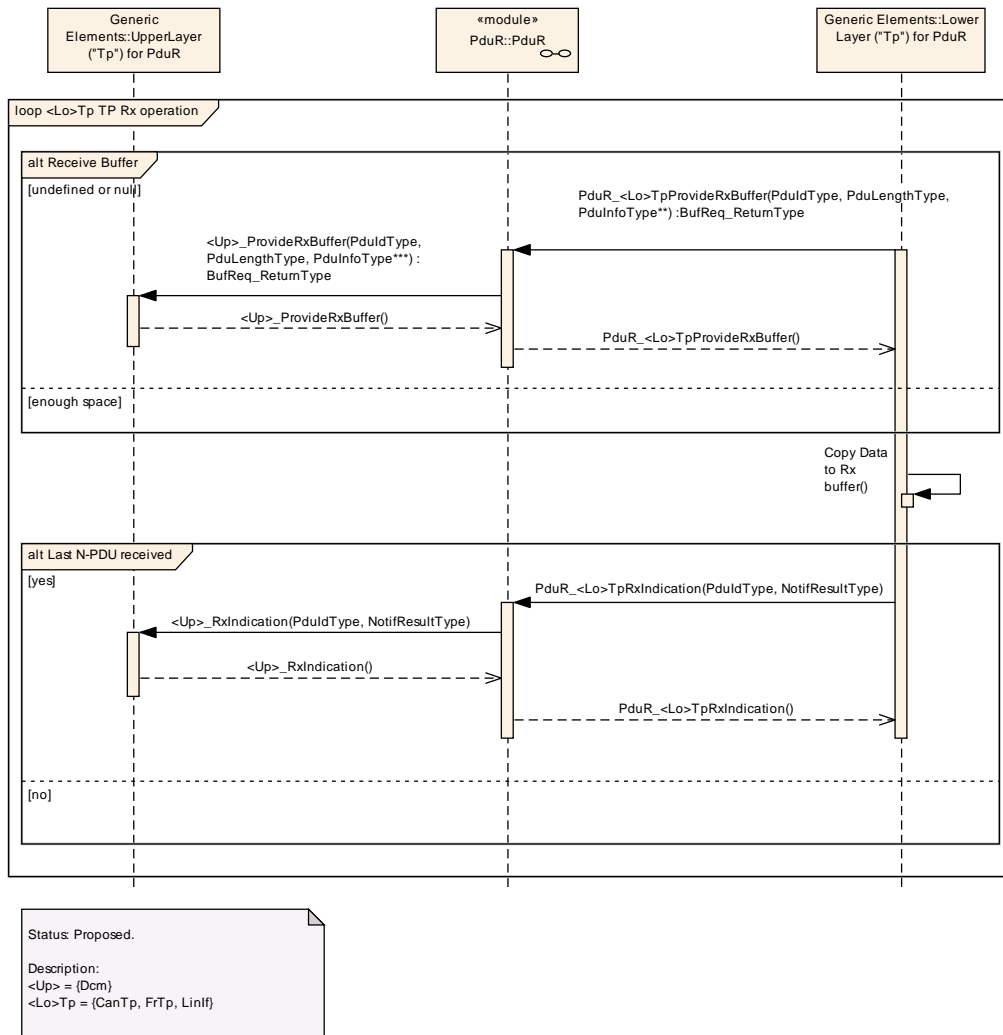
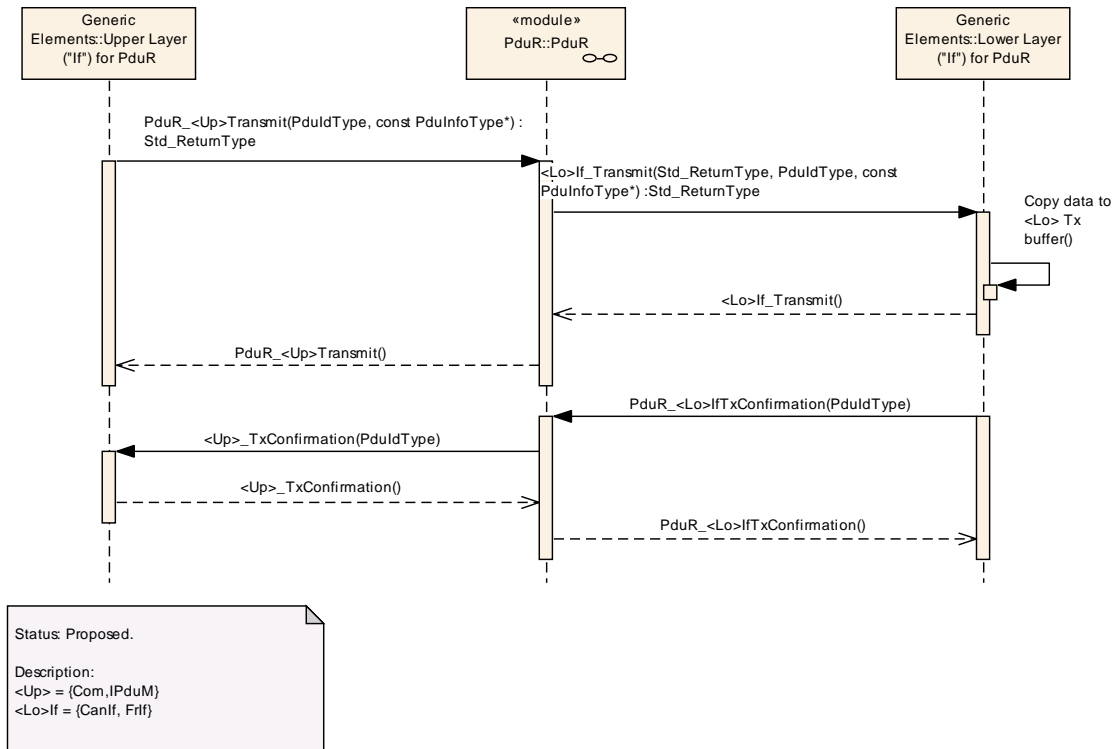


Figure 7: TP-PDU-Rx

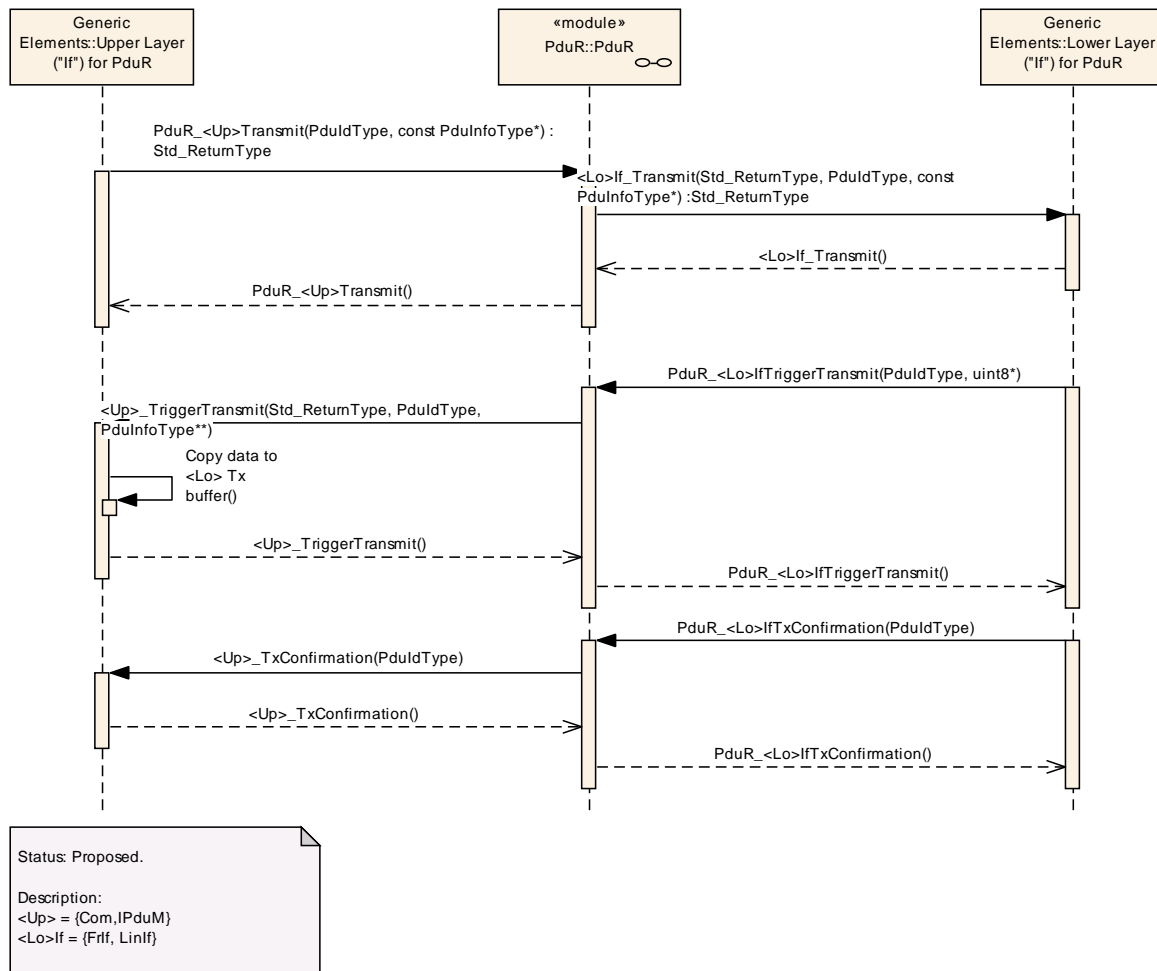
### 9.3 PDU Transmission

The transmission of an I-PDU directly via an interface module (non-TP PDU Tx) is shown by Figure 8 (without trigger transmit) and Figure 9 (with trigger transmit). In the first case the data to be transmitted is provided via the PduInfoPtr parameter of the transmit request. Therefore the data will be copied by the interface module and transmitted on the related bus. If statically configured for the PDU a transmit confirmation is provided.



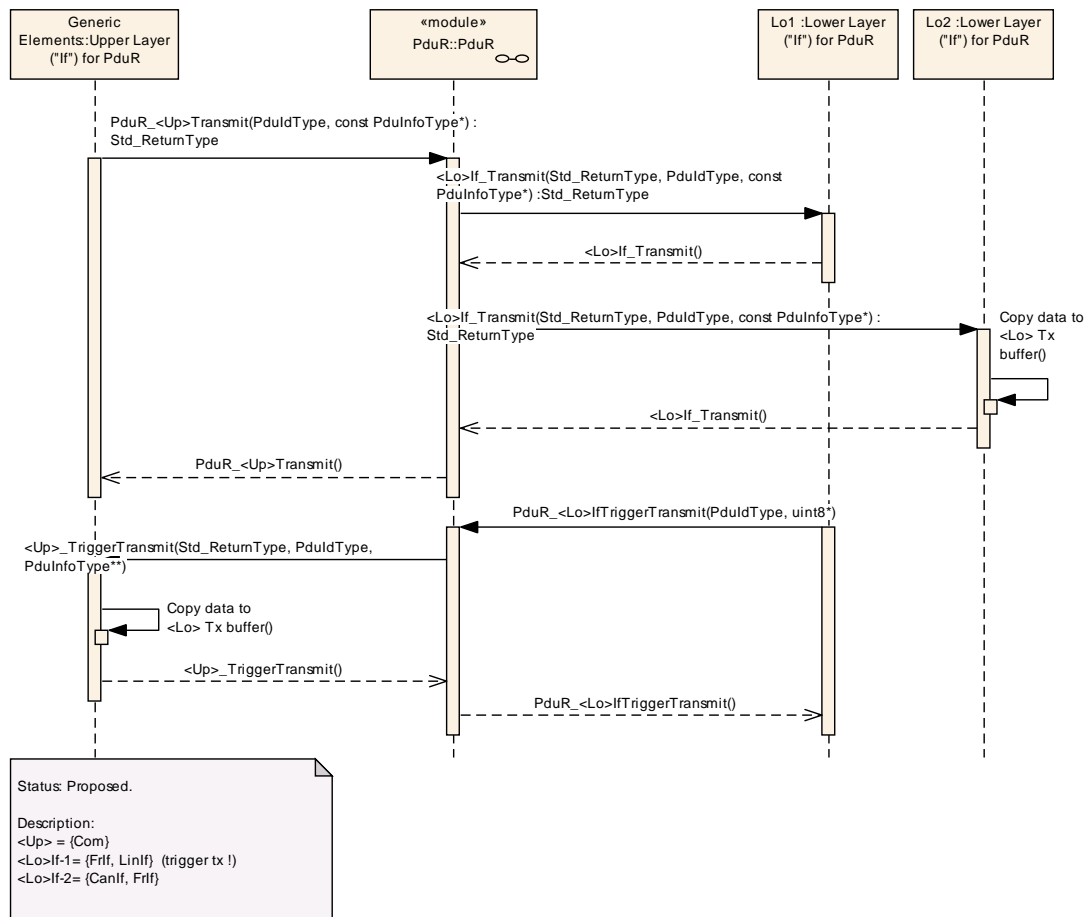
**Figure 8: non TP-PDU-Tx without trigger transmit**

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



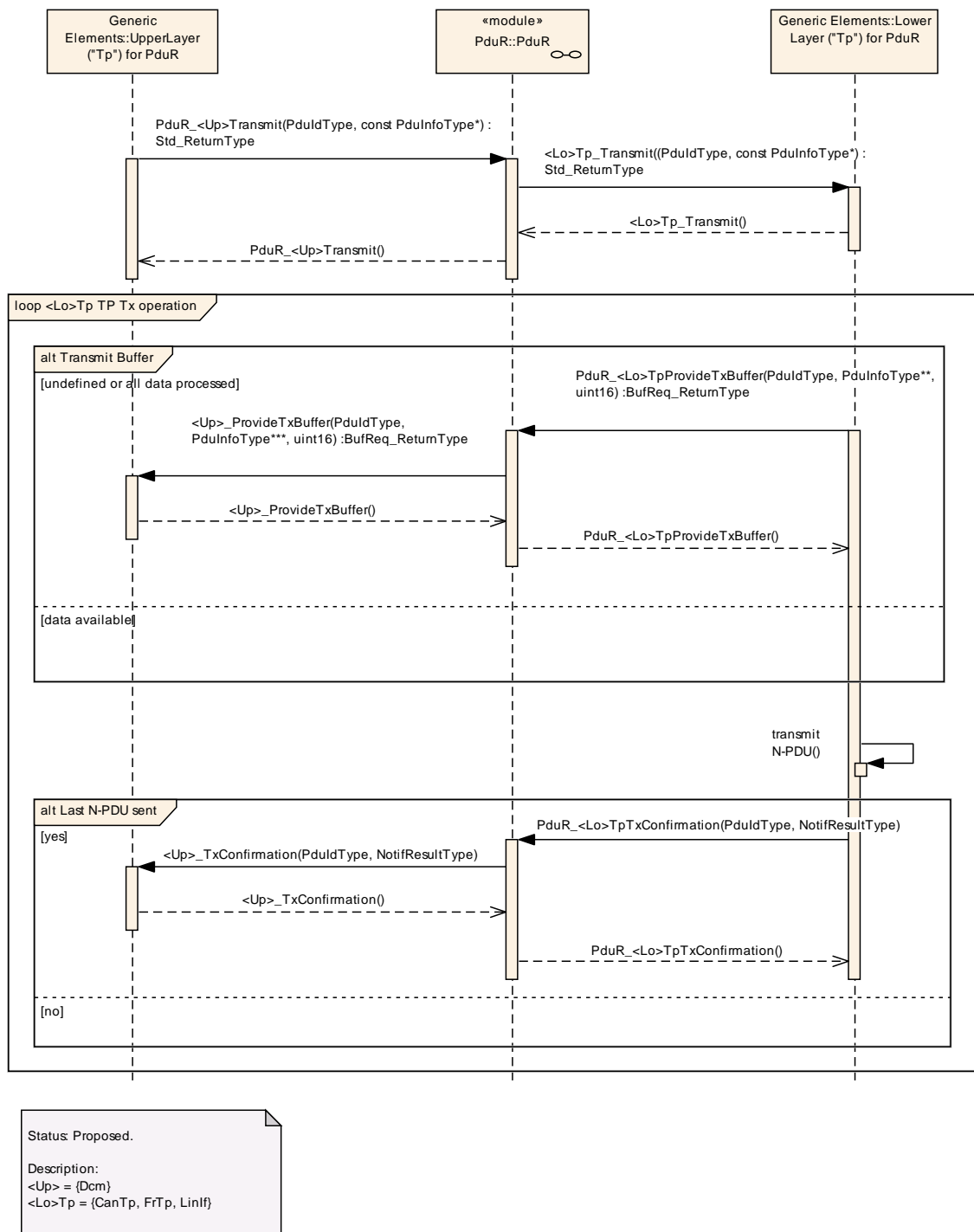
**Figure 9: non-TP-PDU-Tx with trigger transmit**

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



**Figure 10: multicast non-TP PDU Tx**

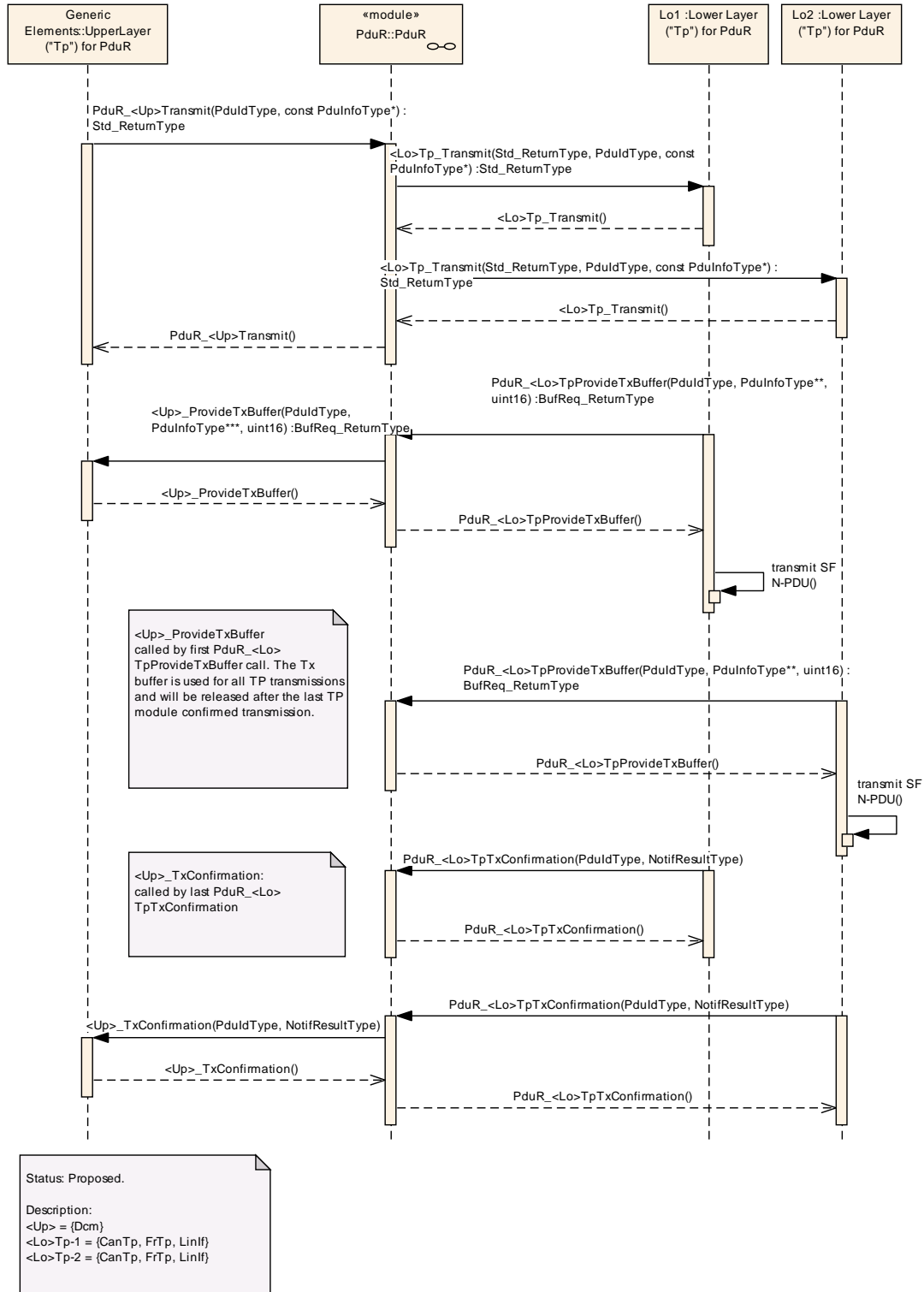
Figure 11 shows the transmission of an I-PDU via a transport protocol module (TP PDU Tx). First the transmit request is forwarded by the PDU Router to the related TP module. Then the TP module executes the loop "TP Tx operation" for each N-PDU transmission. Depending on the status of the transmit buffer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



**Figure 11: TP-PDU-Tx**

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

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



**Figure 12: multicast SF TP-PDU-Tx**

### 9.4 PDU Gateway

Figure 13 and

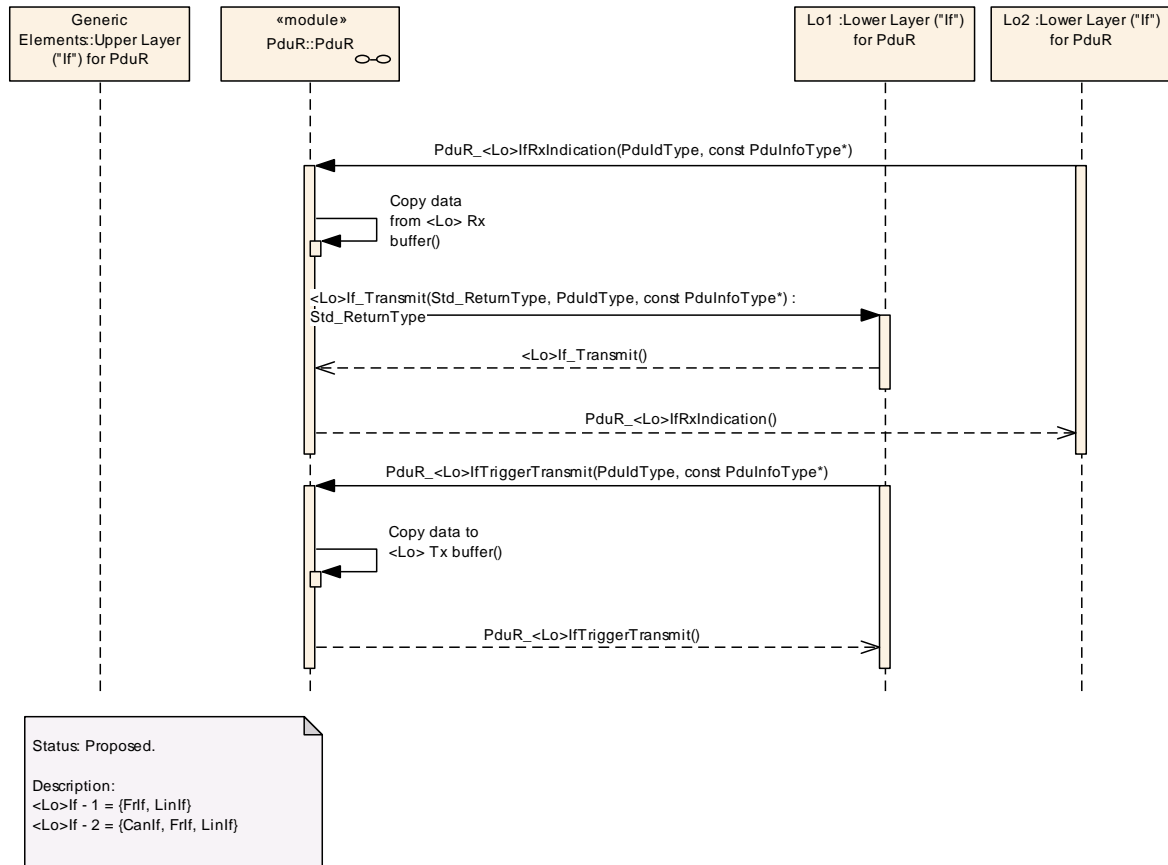
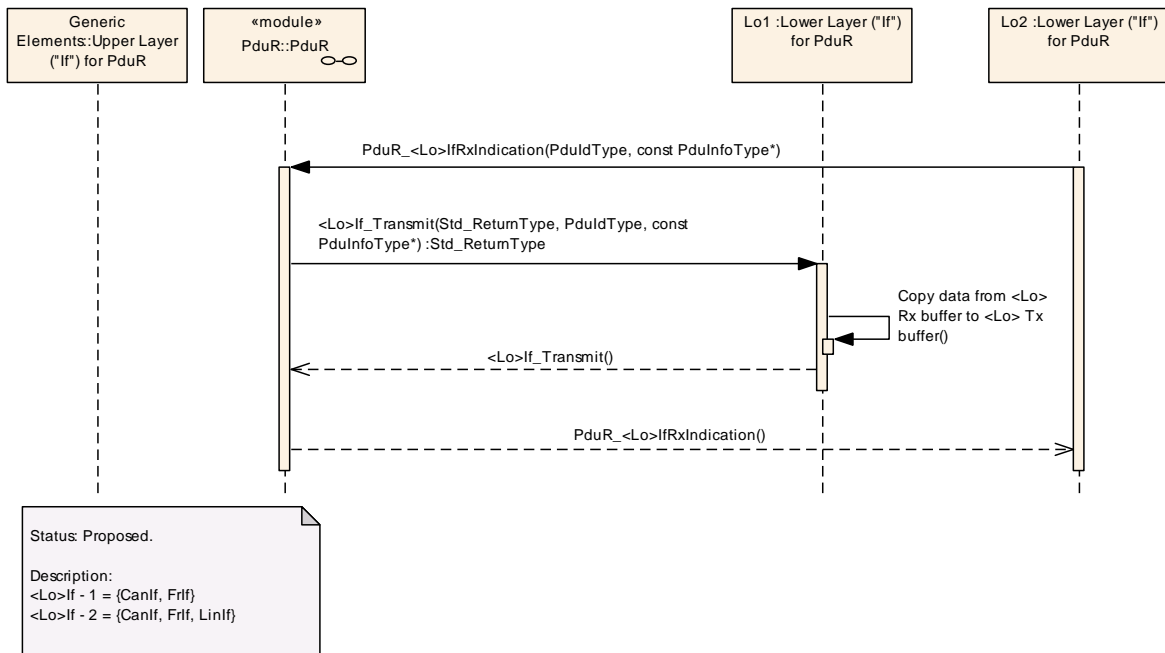


Figure 14 show the PDU Router acting as a direct PDU gateway between two interface modules (non TP PDU Gateway without rate conversion). PDUs received from one bus (interface module 2, rightmost line) shall be forwarded to the other bus (interface module 1). First of all it is shown that no upper layer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

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





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

In the second case

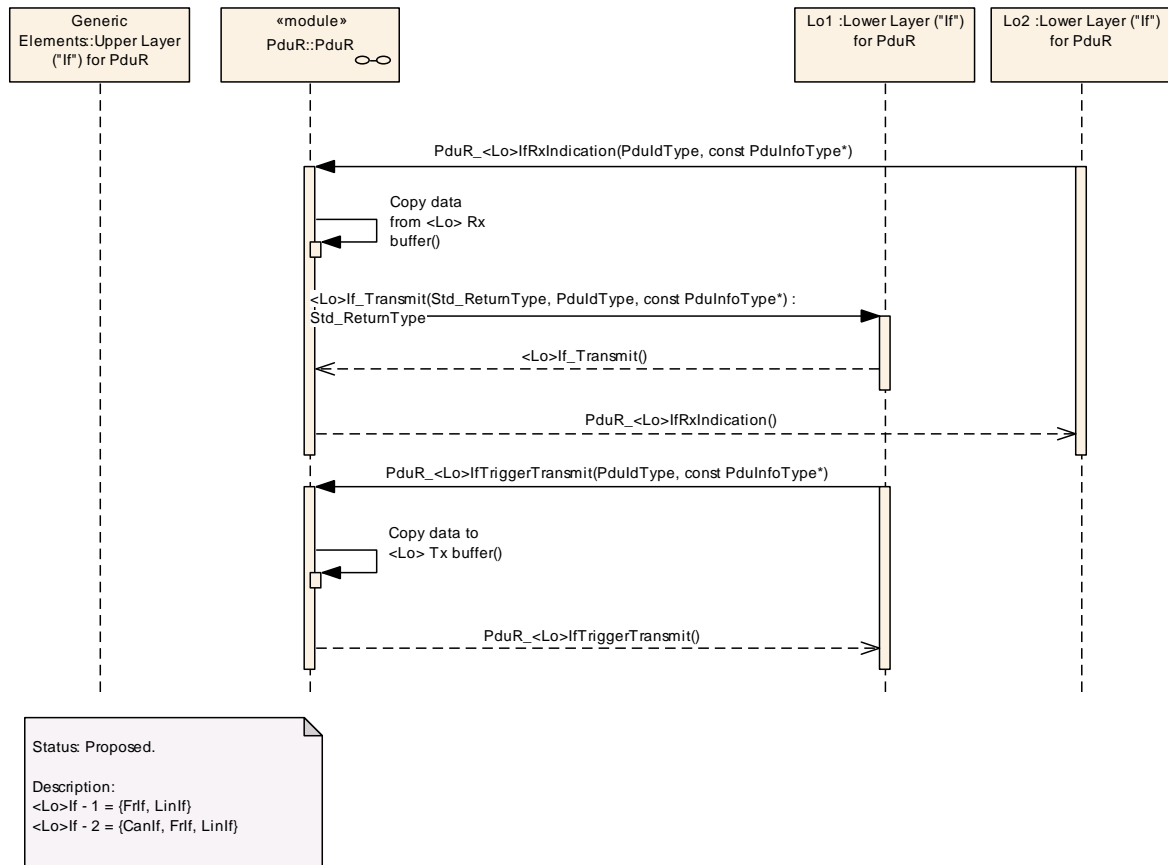
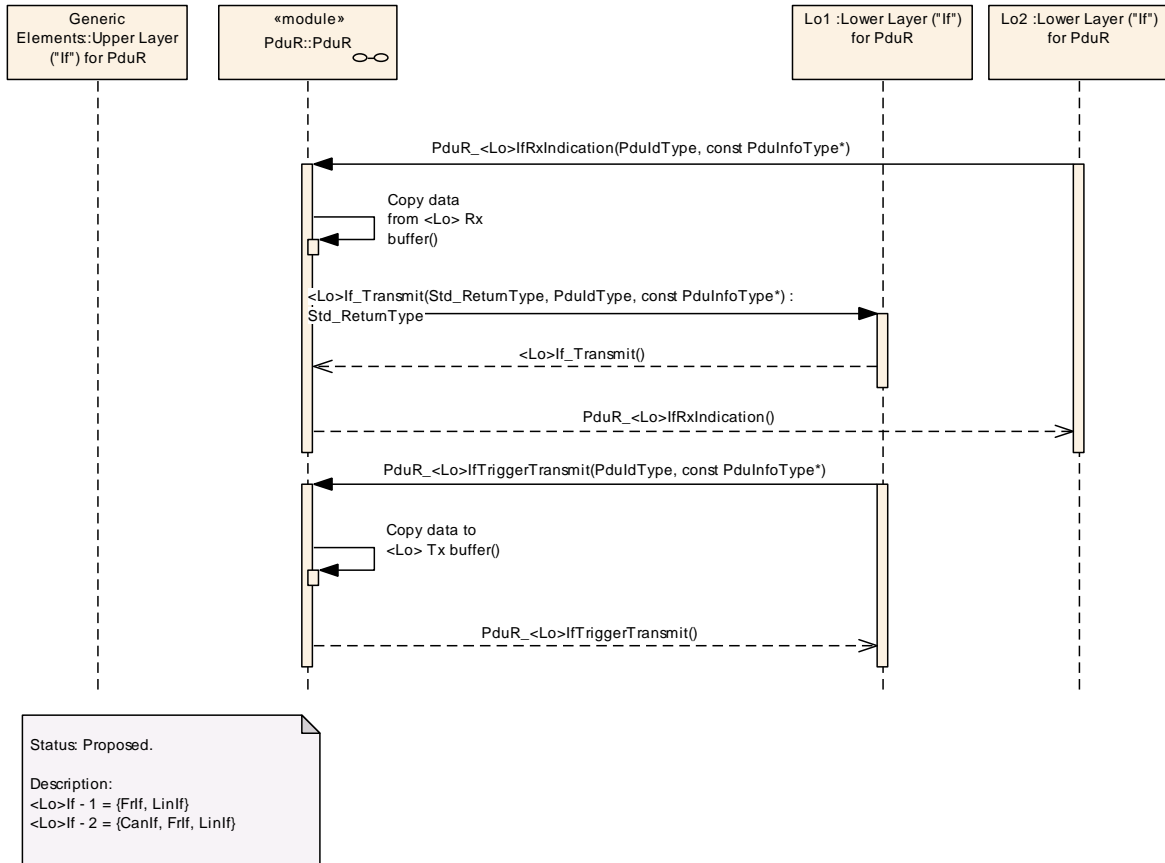
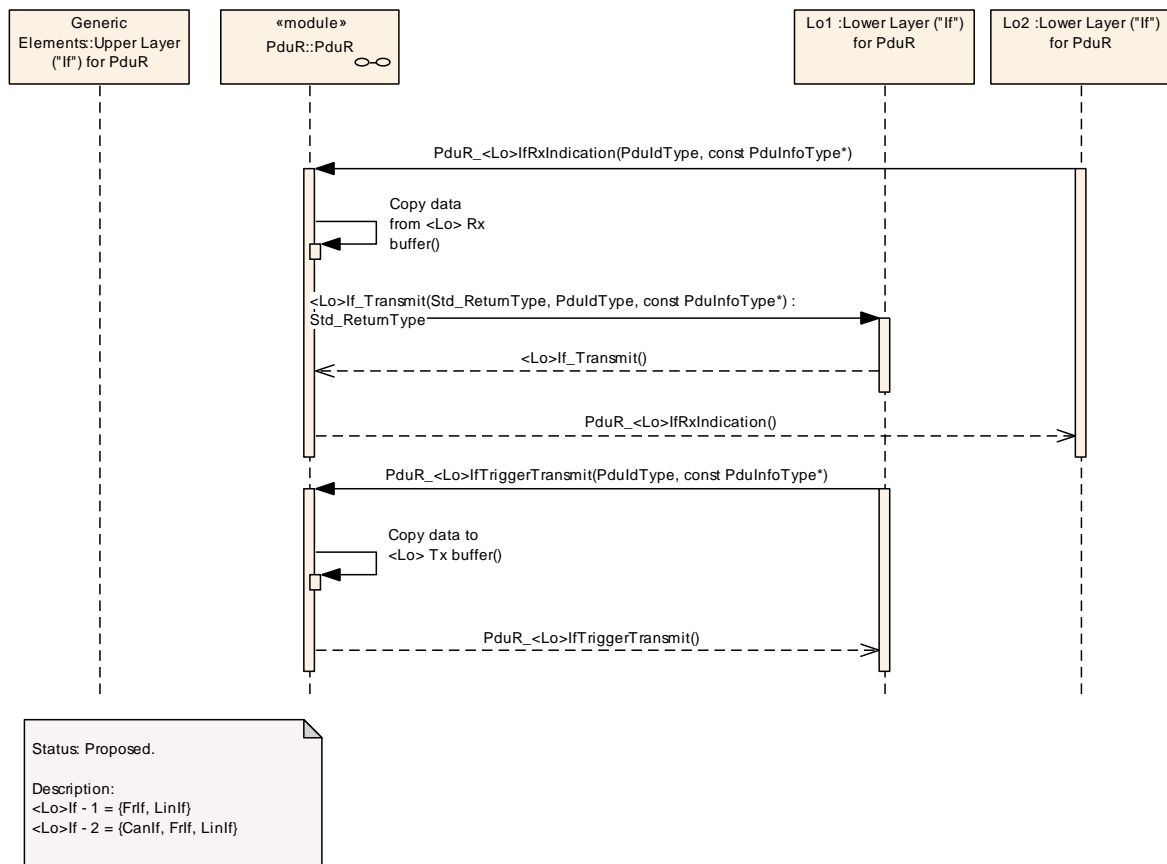


Figure 14) the PDU to be transmitted via interface module 1 is configured to use TriggerTransmit data provision. Therefore the data received from interface module 2 will not be provided as part of the transmit request to interface module 1. It will later be retrieved by interface module 1 via the TriggerTransmit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



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

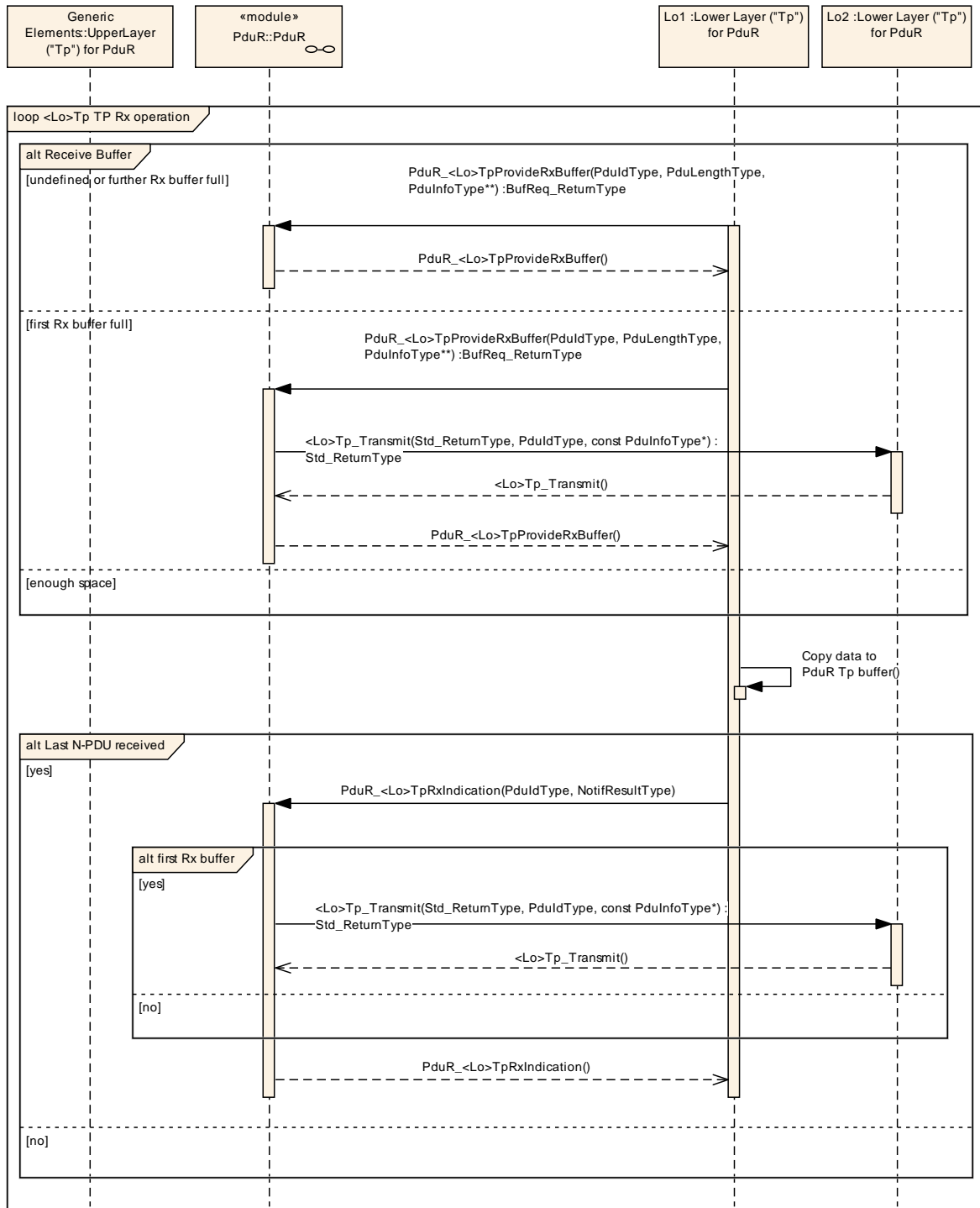
A PDU Gateway between two interface modules with rate conversion is not directly supported by the PDU Router. But as shown by Figure 15 this could be done by AUTOSAR COM. It simply consists of two parts: (1) PDU reception from interface module 2 (cp. Figure 6) and (2) PDU transmission via interface module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



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

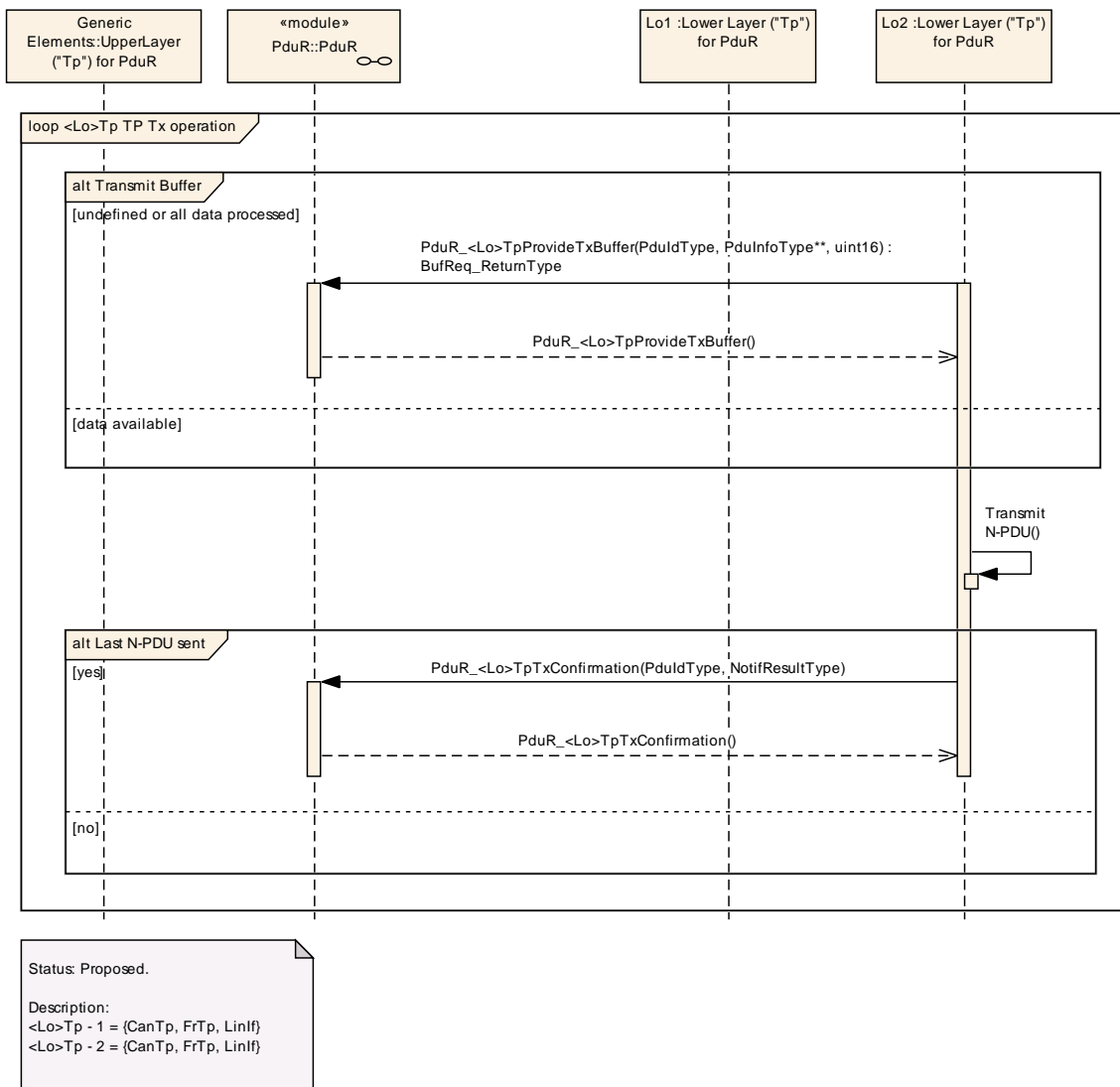
The sequence diagram of a PDU gateway between two TP modules (TP PDU Gateway) is shown by Figure 16 and Figure 17. Data received from one bus (TP module 1) shall be forwarded to another bus (TP module 2, rightmost line).

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



Status: Proposed.  
Description:  
<Lo>Tp - 1 = (CanTp, FrTp, LinIf)  
<Lo>Tp - 2 = (CanTp, FrTp, LinIf)

Figure 16: TP PDU Gateway part 1

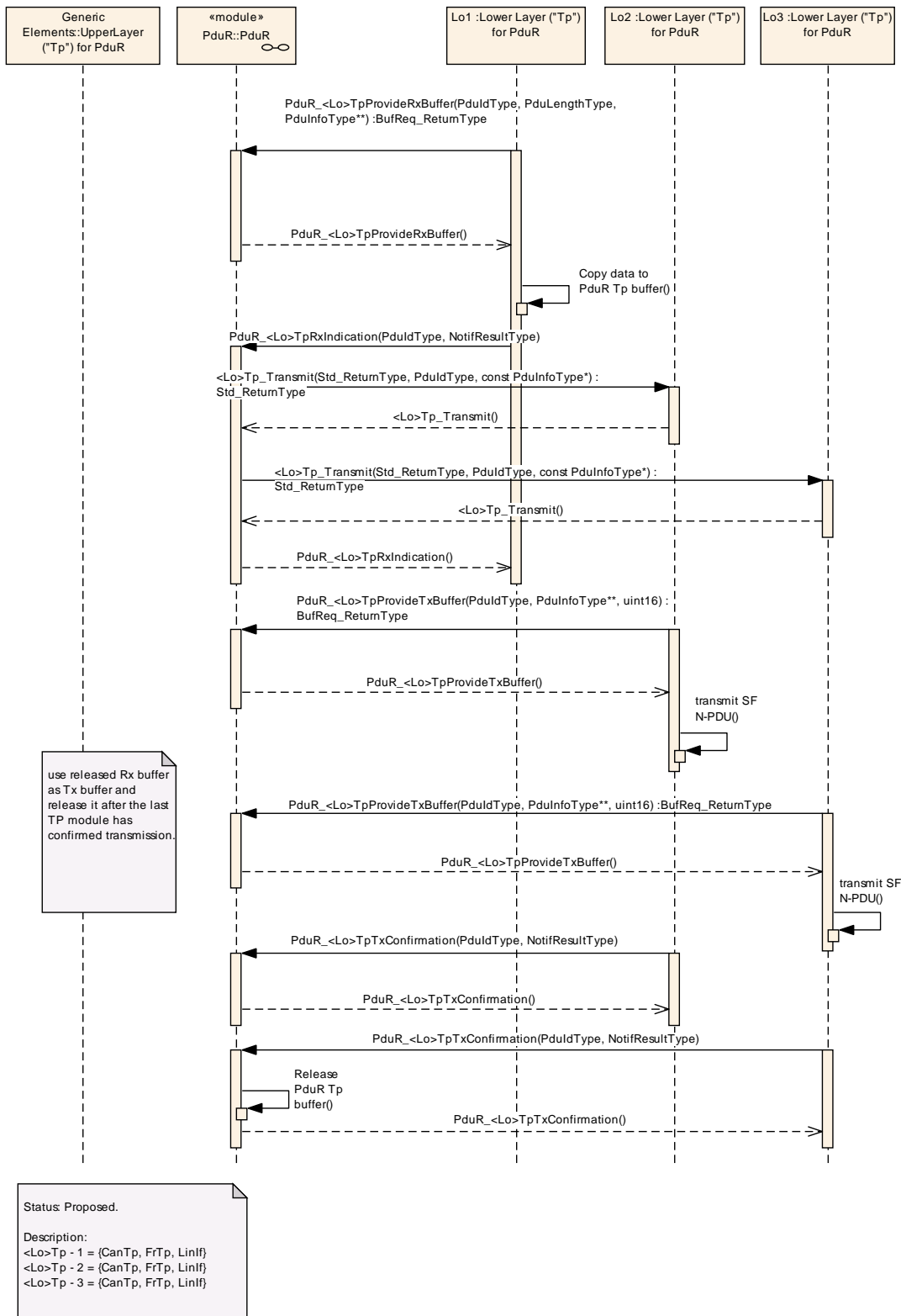


**Figure 17: TP PDU Gateway part 2**

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

Figure 18 shows a TP PDU Gateway with two destination busses (multicast SF TP PDU Gateway). Also for this gateway operation no upper layer module is involved (empty line, leftmost). In contrast to a multiple frame TP reception or transmission no loop operations have to be executed as only a single N-PDU will be received or transmitted (on each bus) respectively. The PDU Router provides the receive buffer when it is requested by the receiving TP module (TP module 1). Within `PduR_<Lo>TpRxIndication` the PDU Router will request a TP transmission at TP module 2 and TP module 3. The released receive buffer will be provided as a

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



**Figure 18: multicast SF TP PDU Gateway**



## 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave chapter 10.1 in the specification to guarantee comprehension.

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

Chapter 10.3 specifies published information of the module PDU Router.

### 10.1 How to read this chapter

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

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

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

#### 10.1.1 Configuration and configuration parameters

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

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

#### 10.1.2 Variants

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

### 10.1.3 Containers

Containers structure the set of configuration parameters. This means:

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

### 10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

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

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

<b>Label</b>	<b>Description</b>
x	The configuration parameter shall be of configuration class <i>Pre-compile time</i> .
--	The configuration parameter shall never be of configuration class <i>Pre-compile time</i> .

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

<b>Label</b>	<b>Description</b>
x	The configuration parameter shall be of configuration class <i>Link time</i> .
--	The configuration parameter shall never be of configuration class <i>Link time</i> .

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

<b>Label</b>	<b>Description</b>
x	The configuration parameter shall be of configuration class <i>Post Build</i> and no specific implementation is required.
L	<i>Loadable</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and only one configuration parameter set resides in the ECU.
M	<i>Multiple</i> - the configuration parameter shall be of configuration class <i>Post Build</i> 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 <i>Post Build</i> .

## 10.2 Containers and configuration parameters

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

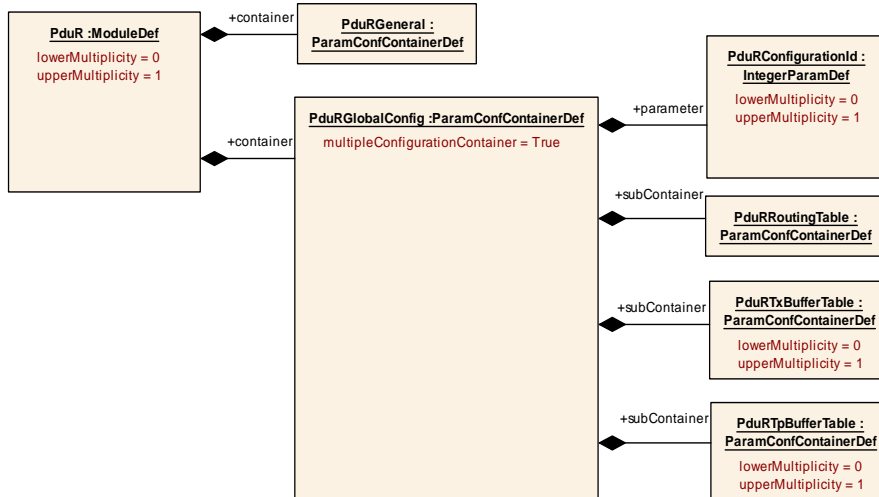


Figure 19: PDU Router Configuration Overview - PduR

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

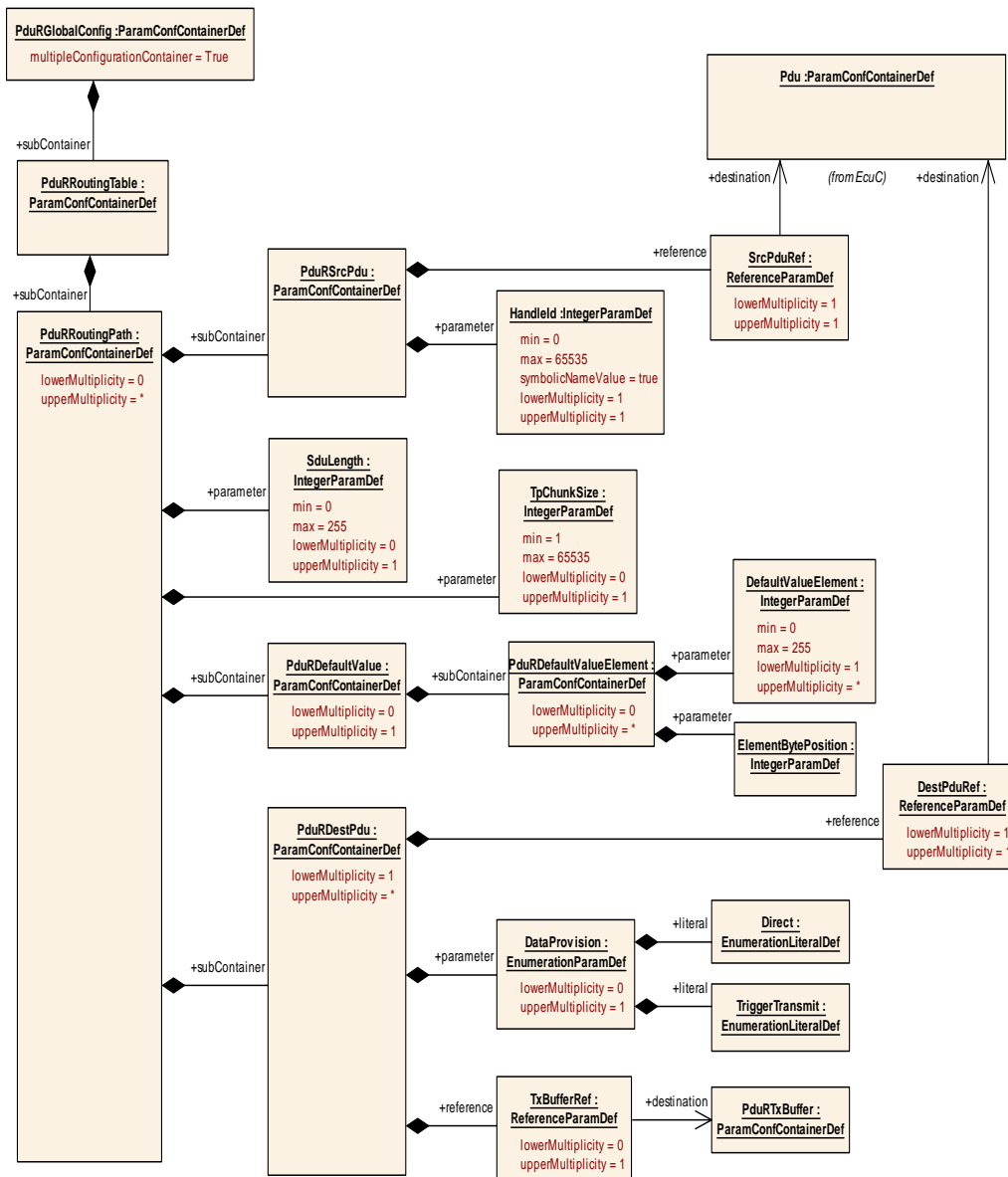


Figure 20: PDU Router Configuration Overview – Routing Paths

### 10.2.1 Variants

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

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

This variant is only possible in zero-cost operation.

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

## 10.2.2 PduR

<b>Module Name</b>	PduR
<b>Module Description</b>	Configuration of the PduR (PDU Router) module.

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

## 10.2.3 PduRGeneral

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

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

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRComSupport {PDUR_COM_SUPPORT}		
<b>Description</b>	Configuration parameter to enable or disable PDU Router support for COM.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants

	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

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

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRFifoTxBufferSupport {PDUR_FIFO_TX_BUFFER_SUPPORT}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRFrTpSupport {PDUR_FRTP_SUPPORT}		

<b>Description</b>	Configuration parameter to enable or disable PDU Router support for FlexRay TP.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

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

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

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRLinTpSupport {PDUR_LINTP_SUPPORT}		
<b>Description</b>	Configuration parameter to enable or disable PDU Router support for LIN TP.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		

<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRMinimumRoutingLoModule {PDUR_MINIMUM_ROUTING_LO_MODULE}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EnumerationParamDef		
<b>Range</b>	CAN_IF	Can Interface	
	CAN_TP	Can TP	
	FR_IF	FlexRay Interface	
	FR_TP	FlexRay TP	
	LIN_IF	Lin Interface	
	LIN_TP	Lin TP	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	X	VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMinimumRoutingLoRxPduId {PDUR_MINIMUM_ROUTING_LO_RXPDUID}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	IntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	X	VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			



<b>SWS Item</b>	--		
<b>Name</b>	PduRMinimumRoutingLoTxPduld {PDUR_MINIMUM_ROUTING_LO_TXPDUID}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	IntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	X	VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMinimumRoutingUpModule {PDUR_MINIMUM_ROUTING_UP_MODULE}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EnumerationParamDef		
<b>Range</b>	COM	COM	
	DCM	DCM	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	X	VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMinimumRoutingUpRxDuld {PDUR_MINIMUM_ROUTING_UP_RXPDUID}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	IntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	X	VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMinimumRoutingUpTxPduld {PDUR_MINIMUM_ROUTING_UP_TXPDUID}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	IntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	X	VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMulticastFromIfSupport {PDUR_MULTICAST_FROMIF_SUPPORT}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMulticastFromTpSupport {PDUR_MULTICAST_FROMTP_SUPPORT}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRMulticastToIfSupport {PDUR_MULTICAST_TOIF_SUPPORT}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
-----------------	----	--	--

<b>Name</b>	PduRMulticastToTpSupport {PDUR_MULTICAST_TOTP_SUPPORT}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	PduRSbTxBufferSupport {PDUR_SB_TX_BUFFER_SUPPORT}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

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

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

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRZeroCostOperation {PDUR_ZERO_COST_OPERATION}		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

**No Included Containers**

#### 10.2.4 PduRTxBufferTable

<b>SWS Item</b>	<b>PDUR243 :</b>		
<b>Container Name</b>	PduRTxBufferTable		
<b>Description</b>	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.		
<b>Configuration Parameters</b>			

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

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
PduRTxBuffer	0..*	This container is a subcontainer of PduRTxBufferTable and specifies a transmit buffer for a non-TP PDU.

### 10.2.5 PduRTxBuffer

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

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

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

**No Included Containers**

### 10.2.6 PduRRoutingTable

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

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

### 10.2.7 PduRRoutingPath

<b>SWS Item</b>	<b>PDUR248 :</b>		
<b>Container Name</b>	PduRRoutingPath		

<b>Description</b>	This container is a subcontainer of PduRRoutingTable and specifies the routing path of a PDU.
<b>Configuration Parameters</b>	

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

<b>SWS Item</b>	--		
<b>Name</b>	TpChunkSize		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	IntegerParamDef		
<b>Range</b>	1 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	--	
	<b>Post-build time</b>	L	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

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

### 10.2.8 PduRSrcPdu

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

<b>SWS Item</b>	--		
<b>Name</b>	HandleId		
<b>Description</b>	PDU identifier assigned by PDU Router.		

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

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

**No Included Containers**

### 10.2.9 PduRDestPdu

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

<b>SWS Item</b>	--		
<b>Name</b>	DataProvision		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EnumerationParamDef		
<b>Range</b>	DIRECT	direct data provision	
	TRIGGER_TRANSMIT	trigger transmit data provision	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	--	
	<b>Post-build time</b>	L	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	DestPduRef		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to Pdu		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	L	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

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

**No Included Containers**

### 10.2.10 PduRDefaultValue

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

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

### 10.2.11 PduRDefaultValueElement

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

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

<b>SWS Item</b>	--		
<b>Name</b>	ElementBytePosition		



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

**No Included Containers**

### 10.2.12 PduRTpBufferTable

<b>SWS Item</b>	<b>PDUR245 :</b>		
<b>Container Name</b>	PduRTpBufferTable{TpBufferTable}		
<b>Description</b>	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.		
<b>Configuration Parameters</b>			

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

**Included Containers**

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

### 10.2.13 PduRTpBuffer

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

<b>SWS Item</b>	--		
<b>Name</b>	Length		
<b>Description</b>	Length of the buffer.		
<b>Multiplicity</b>	1		
<b>Type</b>	IntegerParamDef		
<b>Range</b>	1 .. 65535		

<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	--	
	<b>Post-build time</b>	L	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

**No Included Containers**

### 10.2.14 PduRGlobalConfig

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

<b>SWS Item</b>	--		
<b>Name</b>	PduRConfigurationId {PDUR_CONFIGURATION_ID}		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	IntegerParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	--	
	<b>Post-build time</b>	L	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

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

## 10.3 Published Information

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

The standard common published information like

vendorId (<Module>\_VENDOR\_ID),  
moduleId (<Module>\_MODULE\_ID),  
arMajorVersion (<Module>\_AR\_MAJOR\_VERSION),  
arMinorVersion (<Module>\_AR\_MINOR\_VERSION),  
arPatchVersion (<Module>\_AR\_PATCH\_VERSION),  
swMajorVersion (<Module>\_SW\_MAJOR\_VERSION),  
swMinorVersion (<Module>\_SW\_MINOR\_VERSION),  
swPatchVersion (<Module>\_SW\_PATCH\_VERSION),  
vendorApiInfix (<Module>\_VENDOR\_API\_INFIX)

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

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

## 10.4 Plausibility checks of configuration

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

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

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

(3) If PduRGatewayOperation is disabled, the pre-compile time configuration parameters PduRSbTxBufferSupport, PduRFifoTxBufferSupport, PduRMulticastFromIfSupport and PduRMulticastFromTpSupport must be disabled as well.

(4) TpChunkSize (PduRRoutingTable/PduRRoutingPath) shall not be greater than the length of the largest TP Buffer (PduRTpBufferTable/PduRTpBuffer).

## 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):

<i>ComTxPduld</i>	<i>TargetFctPtr</i>	<i>TargetPduld</i>
0	CanIf_Transmit	0
1	Frf_Transmit	0
2	CanIf_Transmit	1
3	Multilf_Transmit	0
4	Multilf_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 (MultIf\_Transmit) and an additional routing table is used.

Routing table used by MultIf\_Transmit for IfTxPDUs:

<i>Index</i>	<i>MPdul</i>	<i>TargetFctPtr</i>	<i>TargetPdul</i>
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 MPdul. 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 MPdul (e.g. 0 and 2). All subsequent entries with the same MPdul belong to the same multicast request. The execution of a multicast operation ends at an entry with a different MPdul.

Routing table used by PduR\_<Lo>IfTxConfirmation and PduR\_<Lo>IfTriggerTransmit for IfTxPDUs of <Lo>If:

<i>CanIfTxPdul</i>	<i>TargetFctPtr1</i>	<i>TargetPdul</i>
0	Com_TxConfirmation	0
1	Com_TxConfirmation	2
2	NULL	0
3	NULL	0
4	NULL	0
5	NULL	0
6	NULL	0
...	...	...

<i>FrIfTxPdul</i>	<i>TargetFctPtr1</i>	<i>TargetFctPtr2</i>	<i>TargetPdul</i>
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>IfTxPduIs are used by the PDU Router; e.g. FrIfTxPdul = 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 (CanIfTxPdul = 3 and 4, FrIfTxPdul = 2 and 3) or gateway operation (FrIfTxPdul = 4 and 5).

Routing table used by PduR\_<Lo>IfRxIndication for IfRxPDUs received from <Lo>If:

<i>CanIfRxPduld</i>	<i>TargetFctPtr1</i>	<i>TargetPduld</i>
0	Com_RxIndication	0
1	MG_IfRxIndication	0
2	MG_IfRxIndication	1
...	...	...

<i>FrIfRxPduld</i>	<i>TargetFctPtr1</i>	<i>TargetPduld</i>
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:

<i>Index</i>	<i>MG Pduld</i>	<i>TargetFctPtr1</i>	<i>TargetFctPtr2</i>	<i>TargetPduld</i>	<i>SDU length</i>	<i>Buffer Type</i>	<i>TxBufferIdx</i>		
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 MGPduld. 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 MGPduld (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):

<i>DcmTxPduld</i>	<i>TargetFctPtr</i>	<i>TargetPduld</i>
0	CanTp_Transmit	0
1	CanTp_Transmit	1
2	FrTp_Transmit	0
3	MultiTp_Transmit	0
...	...	...

Routing table used by MultiTp\_Transmit for TpTxPDUs:

<i>Index</i>	<i>MPduld</i>	<i>TargetFctPtr</i>	<i>TargetPduld</i>
0	0	FrTp_Transmit	1
1	0	CanTp_Transmit	2
2	2	NULL	0

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

<i>CanTpTxPduld</i>	<i>TargetFctPtr1</i>	<i>TargetFctPtr2</i>	<i>Target Pduld</i>	<i>MultiTp</i>
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
...	...	...	...	...

<i>FrTpTxPduld</i>	<i>TargetFctPtr1</i>	<i>TargetFctPtr2</i>	<i>Target Pduld</i>	<i>MultiTp</i>
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 (CanTpTxPduld = 2) and the second entry of the second table (FrTPTxPduld = 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:

<b>CanTpRxPduld</b>	<b>TargetFctPtr1</b>	<b>TargetFctPtr2</b>	<b>TargetPduld</b>
0	Dcm_ProvideRxBuffer	Dcm_RxIndication	0
...	...		...

<b>FrTpRxPduld</b>	<b>TargetFctPtr1</b>	<b>TargetFctPtr2</b>	<b>TargetPduld</b>
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:

<b>Index</b>	<b>MG Pduld</b>	<b>TargetFctPtr1</b>	<b>TargetFctPtr2</b>	<b>TargetFctPtr3</b>	<b>Target Pduld</b>		
0	0	NULL	NULL	FrTp_Transmit	2		G
1	1	NULL	NULL	CanTp_Transmit	3	M	G
2	1	Dcm_ProvideRxBuffer	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 2

### 11.1 Deleted SWS Items

<i>SWS Item</i>	<i>Rationale</i>
PDUR321	Result of Bug#18912
PDUR322	Result of Bug#18912
PDUR323	Result of Bug#18912
PDUR165	Result of Bug#18912
PDUR426	Variant 2 removed as per Bug#20777

### 11.2 Replaced SWS Items

<i>SWS Item of Release 1</i>	<i>replaced by SWS Item</i>	<i>Rationale</i>
PDUR291	<a href="#">PDUR425</a> , <a href="#">PDUR426</a> , <a href="#">PDUR427</a>	Definitions separated to single requirements

### 11.3 Changed SWS Items

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

## 11.4 Added SWS Items

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

## 12 Changes during SWS Improvements by Technical Office

### 12.1 Deleted SWS Items

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

### 12.2 Replaced SWS Items

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

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

### 12.3 Changed SWS Items

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

<b>SWS Item</b>	<b>Rationale</b>

### 12.4 Added SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">PDUR333</a>	UML model linking of imported types
<a href="#">PDUR334</a>	UML model linking of PduR_Init
<a href="#">PDUR338</a>	UML model linking of PduR_GetVersionInfo
<a href="#">PDUR339</a>	Requirement on PduR_GetVersionInfo
<a href="#">PDUR340</a>	Requirement on PduR_GetVersionInfo
<a href="#">PDUR341</a>	UML model linking of PduR_GetConfigurationId
<a href="#">PDUR342</a>	Requirement on PduR_GetConfigurationId
<a href="#">PDUR343</a>	UML model linking of PduR_CanIfRxIndication
<a href="#">PDUR344</a>	Requirement on PduR_CanIfRxIndication
<a href="#">PDUR345</a>	Requirement on PduR_CanIfRxIndication
<a href="#">PDUR346</a>	UML model linking of PduR_CanIfTxConfirmation
<a href="#">PDUR347</a>	Requirement on PduR_CanIfTxConfirmation
<a href="#">PDUR348</a>	Requirement on PduR_CanIfTxConfirmation
<a href="#">PDUR349</a>	Requirement on PduR_CanIfTxConfirmation
<a href="#">PDUR350</a>	UML model linking of PduR_CanTpProvideRxBuffer
<a href="#">PDUR351</a>	Requirement on PduR_CanTpProvideRxBuffer
<a href="#">PDUR352</a>	Requirement on PduR_CanTpProvideRxBuffer
<a href="#">PDUR354</a>	Requirement on PduR_CanTpRxIndication
<a href="#">PDUR355</a>	Requirement on PduR_CanTpRxIndication
<a href="#">PDUR356</a>	UML model linking of PduR_CanTpProvideTxBuffer
<a href="#">PDUR357</a>	Requirement on PduR_CanTpProvideTxBuffer
<a href="#">PDUR358</a>	Requirement on PduR_CanTpProvideTxBuffer
<a href="#">PDUR360</a>	Requirement on PduR_CanTpTxConfirmation
<a href="#">PDUR361</a>	Requirement on PduR_CanTpTxConfirmation
<a href="#">PDUR362</a>	UML model linking of PduR_FrlfRxIndication
<a href="#">PDUR363</a>	Requirement on PduR_FrlfRxIndication
<a href="#">PDUR364</a>	Requirement on PduR_FrlfRxIndication
<a href="#">PDUR365</a>	UML model linking of PduR_FrlfTxConfirmation

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