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

Document Version	3.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
02.11.2011	3.2.0	AUTOSAR Administration	• clarifications regarding non-TP PDU routing • new feature: non-TP PDU routing idependent of the Pdu lengh • FIFO handling for non-TP PDU routing clarified / improved • Service ID's for generic serivices introduced • clarification regarding multicast routing of TP-PDU's • DEM error reporting removed
26.10.2010	3.1.0	AUTOSAR Administration	• Introduced new version check • Added Std_ReturnType to PduR_<Lo>TriggerTransmit • Added functionality of PduR_<LoTp>CopyTxData when TsSduLength is zero

11.12.2009	3.0.0	AUTOSAR Administration	• The PDU Router module is made generic to allow any combination of busses, TPs and upper modules. The upper and lower modules are modeled generic and handled by the configuration. • The Transport Protocol API has been redesgined. Compatibility between TP in AR3.x and AR4.0 is described. • Cancel transmission of communication interface I-PDUs has been added. • Cancel reception of Tranport Protocol I-PDUs has been added. • Change parameter of Transport Protocol parameters has been extended. • Legal disclaimer revised
23.08.2009	2.2.2	AUTOSAR Administration	Legal disclaimer revised
22.01.2008	2.2.1	AUTOSAR Administration	Correction figure 3
13.11.2007	2.2.0	AUTOSAR Administration	• Variants have been renamed. • New Callbacks PduR_LinTpChangeParameterConfirmation, PduR_FrTpChangeParameterConfirmation • PduR_CanTpChangeParameterConfirmation has been added. • New API's PduR_ChangeParameterRequest, PduR_CancelTransmitRequest has been added • New Typedefines PduR_ParameterValueType, PduR_CancelReasonType has been added • Document meta information extended • Small layout adaptations made
31.01.2007	2.1.0	AUTOSAR Administration	• Clarifications added (FIFO, TxConf, ...) • Unnecessary development errors removed • SchM_PduR.h and MemMap.h added • Corrections of configuration parameters • More details in Chapter 13 • Legal disclaimer revised • Release Notes added • "Advice for users" revised • "Revision Information" added

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

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

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

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

- Communication interface modules (e.g. Com, LinIf, CanIf, CanNm, FrIf and FrNm)
- Transport Protocol modules (e.g. J1939, LinTp [part of LinIf], CanTp, FrTp)

The routing of I-PDUs are made statically by the I-PDU identifier, no I-PDUs are routed dynamically in run-time or dependent on contents.

The location of the modules can be upper module (e.g. DCM, COM, IpduM) or a lower module (CanIf, FrIf, LinTp, IpduM, CanNm, FrNm). Note that the IpduM is listed as upper and lower module because it has two different roles (upper: Communication between COM module and IpduM module, lower: communication between IpduM module and communication interface module)

The PDU Router module is based on a generic approach of interfaced modules. The module that is interfaced is configured in the PDU Router module configuration. The listed modules in parenthesis in the previous paragraph are just examples, and not an exhaustive list. The PDU Router can be easily configured to support other upper and lower layer modules.

The list of users of the PDU Router module is not fixed. The most common combination of upper and lower layer pairs is listed below:

- AUTOSAR Diagnostic Communication Manager (DCM) and Transport Protocol modules
- AUTOSAR COM and communication interface modules, Transport protocol modules or I-PDU Multiplexer
- I-PDU Multiplexer and communication interface modules

1.1 AUTOSAR architecture

The PDU Router module is a central module in the AUTOSAR communication structure [1]. The Figure 1 gives an overview of the AUTOSAR communication structure.

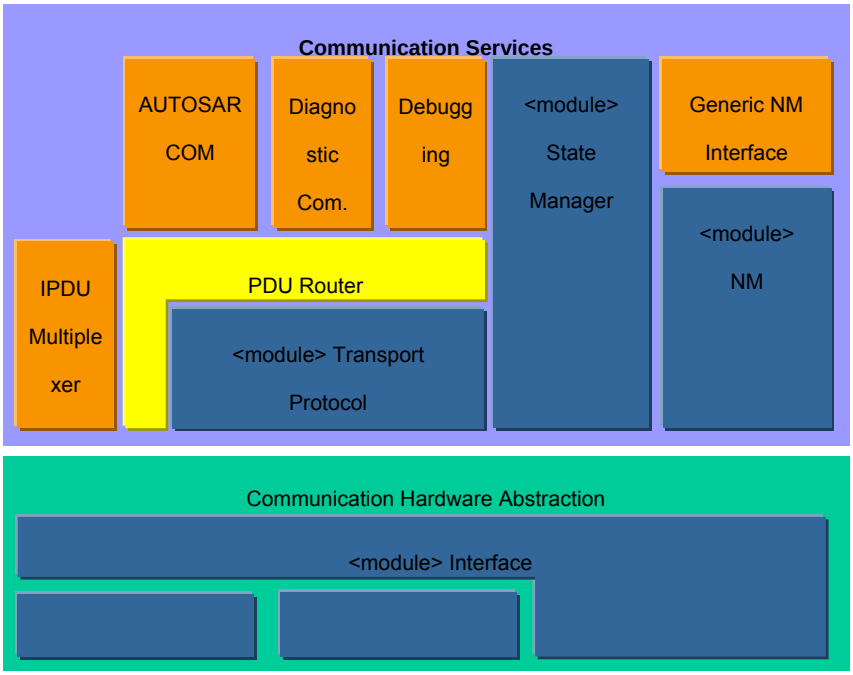

Figure 1: Communication Structure

1.2 PDU Router module function overview

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.

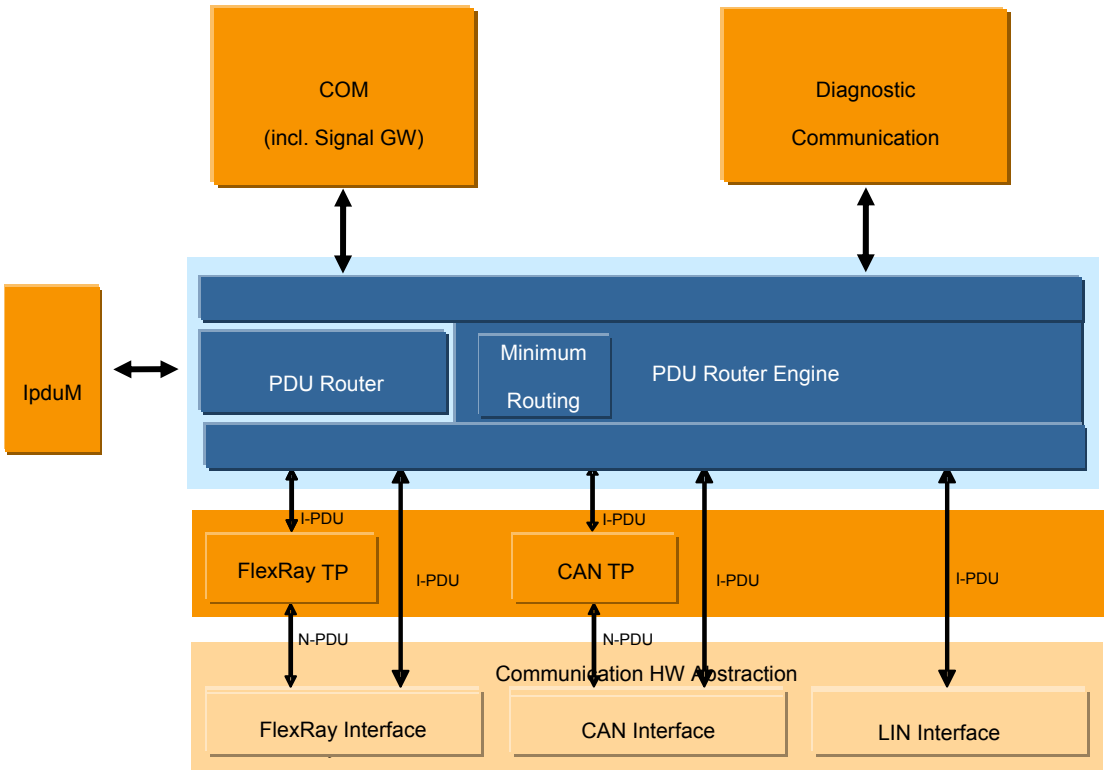

Figure 2: Detailed PDU Router Structure showing FlexRay, CAN and LIN

The PDU Router module mainly consists of two parts:

- The **PDU Router routing tables**: static routing tables describing the routing attributes for each I-PDU to be routed. The routing tables can be (if supported) updated post-build loadable in the programming state of the ECU or selected when initializing the PDU router by post-build selectable (see section 10.1.2).

- The **PDU Router Engine**: the actual code performing routing actions according to the PDU Router routing tables. The router engine has to deal with:
  - Route the I-PDU from source to destination(s).
  - Translating the source I-PDU ID to the destination (e.g. PduR_Transmit to CanIf_Transmit, PduR_CanIfTxConfirmation to Com_TxConfirmation).

Additionally the PDU Router Engine provides a **minimum routing** capability that is

configured at pre-compile or link-time. 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 or not programmed. The minimum routing settings are separated from the PDU Router routing tables and cannot be changed after build-time.

1.3 I-PDU handling

I-PDUs are identified by static I-PDU IDs. The PDU Router module determines the destination of an I-PDU by using the I-PDU ID in a static configuration table. I-PDUs are used for the data exchange of the modules directly above the PDU Router module, e.g. the COM module and DCM module. 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 may start before the full I-PDU is received (“gatewaying-on-the-fly”).

The I-PDU ID is set in the configuration that also implements the API. This will allow an efficient implementation of look-up tables in each module receiving an I-PDU ID (e.g. the PDU Router module’s configuration contains the I-PDU ID for the PduR_CanIfTxConfirmation).

The PDU Router module can:

- Singlecast (1:1) an I-PDU from a local module to a communication interface module.
- Multicast (1:n) an I-PDU from a local module to communication interfaces.
- Singlecast (1:1) an I-PDU (Single Frame (SF) or Multiple frames (FF and CFs)) from a local module to a transport protocol module.
- Multicast (1:n) an I-PDU (Single Frame (SF)) from a local module to transport protocol module(s).
- Gateway (1:1) an I-PDU from a communication interface module to a communication interface module.
- Gateway (1:n) an I-PDU from a communication interface module to communication interface modules.
- Gateway (1:1) an I-PDU (Single Frame (SF) or Multiple frames (FF and CFs)) from a transport protocol module to a transport protocol module.
- Gateway (1:n) an I-PDU (Single Frame (SF)) from a transport protocol module to transport protocol module(s).

For Network Management data exchange the PDU Router module is bypassed.

2 Acronyms and abbreviations

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

Acronym:	*Description:*
Upper Layer Modules (Up)	Modules above the PDU Router. This layer usually includes COM and Diagnostic Communication Manager (DCM). The upper layer modules are configured in the configuration.
Lower Layer Modules (Lo)	Modules below the PDU Router. This layer may include CAN, LIN, FlexRay, Ethernet communication interface modules and the respective TP modules. Modules used are configured in the configuration
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.
gatewaying-on-the-fly	Gateway capability; routing between two TP modules where forwarding of data is started (when a specified threshold is reached) before all data have been received. If larger amount of data is transported between two interfaces it is desirable to be able to start the transmission on the destination network before receiving all data from the source network. This saves memory and time.
multicast operation	Simultaneous transmission of PDUs to a group of receivers, i.e. 1:n routing.
data provision	Provision of data to interface modules. (a) direct data provision: data to be transmitted are provided directly at the transmit request. The destination communication interface may behave in two ways, either copy the data directly or defer the copy to a trigger transmit. (b) trigger transmit data provision: data to be transmitted are not provided at the transmit request, but will be retrieved by the interface module via a callback function

Abbreviation:	*Description:*
I-PDU ID	PDU Identifier
I-PDU	Interaction Layer PDU. An I-PDU consists of data (buffer), length and I-PDU ID. The PDU router will mainly route I-PDUs (exception is routing-on-the-fly)
N-PDU	Network Layer PDU. Used by transport protocol modules to fragment an I-PDU
L-PDU	Data Link Layer PDU. One or more I-PDUs are packed into one L-PDU. The L-PDU is bus specific, e.g. CAN frame.
SF	Single Frame, Transport Protocol term

Abbreviation:	*Description:*
FF	First Frame, Transport Protocol term
CF	Consecutive Frame, Transport Protocol term
PDU	Protocol Data Unit
BSW	Basic Software
<SrcLo>	Lower layer communication interface module acting as a source of the I-PDU. The SrcLo is always one.
<DstLo>	Lower layer communication interface module acting as a destination of the I-PDU. The DstLo may by one to many.
<SrcLoTp>	Lower layer transport protocol module acting as a source of the I-PDU. The SrcLoTp is always one.
<DstLoTp>	Lower layer transport protocol module acting as a destination of the I-PDU. The DstLoTp may by one to many.
<Lo>	Lower layer communication interface module
<Up>	Upper layer module
<LoTp>	Lower layer transport protocol module
<module>	Any type of module <...>

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf,

[2] Requirements on Gateway,
AUTOSAR_SRS_Gateway.pdf

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

[4] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[5] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[6] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf

[7] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[8] Specification of I-PDU Multiplexer
AUTOSAR_SWS_IPDUMultiplexer.pdf

[9] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[10] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

4 Constraints and assumptions

4.1 Limitations

The PDU Router module does **not**:

1. Have mechanisms for signal extraction or conversion.

2. Have mechanisms for data integrity checking (like checksums).

3. Change or modify the I-PDU.

4. Make any PDU payload dependent routing decisions.

5. Support routing between TP modules and communication interface modules or vice versa.

6. Support 1:n routing of I-PDUs which are sent or received via a TP module and require multiple N-PDUs for transmission.

7. 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. the COM module).

4.1.1 Limitations on supported functionality

The PDU Router module supports fan-out of I-PDUs transmitted from a local module (e.g. COM module). There are some limitations if the I-PDU will be transmitted to more than one destination (i.e. fan-out 1:n; n>1). This limitation is because the COM module is not aware how many destinations there are:

- Update bits will not work
- I-PDU sequence counter will not work
- The tx confirmation will be handled in the way that the local module (e.g. COM module) will be informed when all destinations has confirmed the transmission. This means that deadline monitoring is made on all I-PDUs (i.e. it is no difference if one I-PDU was erroneous or if all where erroneous).

Note that above limitations are not set as requirements in since they are not functionality provided by the PDUR module. But implication of the use of the PDUR module will affect these functionalities.

4.2 Applicability to car domains

The PDU Router is used in all ECUs where communication is necessary.

The PDU Router module has not been specified to work with 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>_Transmit (e.g. CanIf_Transmit, FrIf_Transmit, LinIf_Transmit)
- <Lo>_CancelTransmit (e.g. CanIf_CancelTransmit, FrIf_CancelTransmit, LinIf_CancelTransmit)

Transport Protocol Modules:

- <LoTp>_Transmit (e.g. CanTp_Transmit, FrTp_Transmit, LinTp_Transmit)
- <LoTp>_CancelTransmit (e.g. CanTp_CancelTransmit, FrTp_CancelTransmit, LinTp_CancelTransmit)

Upper layer modules which use transport protocol modules:

- <Up>_StartOfReception (e.g. Dcm_StartOfReception)
- <Up>_CopyRxData (e.g. Dcm_CopyRxData)

Upper layer modules which process I-PDUs originating from communication interface modules:

- <Up>_RxIndication (e.g. Com_RxIndication),
- <Up>_TxConfirmation (e.g. Com_TxConfirmation),
- <Up>_TriggerTransmit (e.g. Com_TriggerTransmit)

5.1 File structure

5.1.1 Code file structure

The code file structure is not defined within this specification completely. However to allow integration to other modules the following structure is needed.

[PDUR226] ⌈The code-file structure shall include the following files named:

- PduR_Cfg.c – for pre-compile time configurable parameters,

- PduR_PBcfg.c – for post build time configurable parameters. ⌋(BSW00345, BSW00380, BSW00419, BSW00381)

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. ⌋( )

[PDUR216] ⌈The PDU Router module shall provide the functions used by the different modules in separate header files. ⌋( )

Example: If CanIf, CanTp and FrIf are used then the PDU Router module shall provide PduR_CanIf.h, PduR_CanTp.h and PduR_FrIf.h

[PDUR132] ⌈The include file structure regarding the specifics of the PDU Router module shall be constructed as following list:

- `PduR.h` shall include `PduR_Types.h`, `MemMap.h`, `PduR_Cfg.h`, `PduR_Lcfg.h` and `PduR_PBcfg.h`
- `PduR_Cfg.h` shall include the `<module>_PduR.h` and `<module>_Cbk.h`. The `<module>` is set by the configuration. The `<module>_PduR.h` included the APIs for the interacted modules, and `<module>_Cbk.h` the callbacks.
- The PduR implementation shall include `Dem.h`
- The PduR implementation shall include `Det.h` if the related pre-compile time configuration parameter is enabled.

The include relations for the PDU Router module is depicted in Figure 3. ⌋(BSW00415)

[PDUR0762] ⌈All PDU Router header files shall contain a software and specification version number. ⌋(BSW00379, BSW003)

[PDUR0748] ⌈The following header files shall include the PDU Router module configuration definitions:

- PduR_Cfg.h shall include all pre-compile parameters

- PduR_PBcfg.h shall include all post-build parameters⌋(BSW00345, BSW00412)

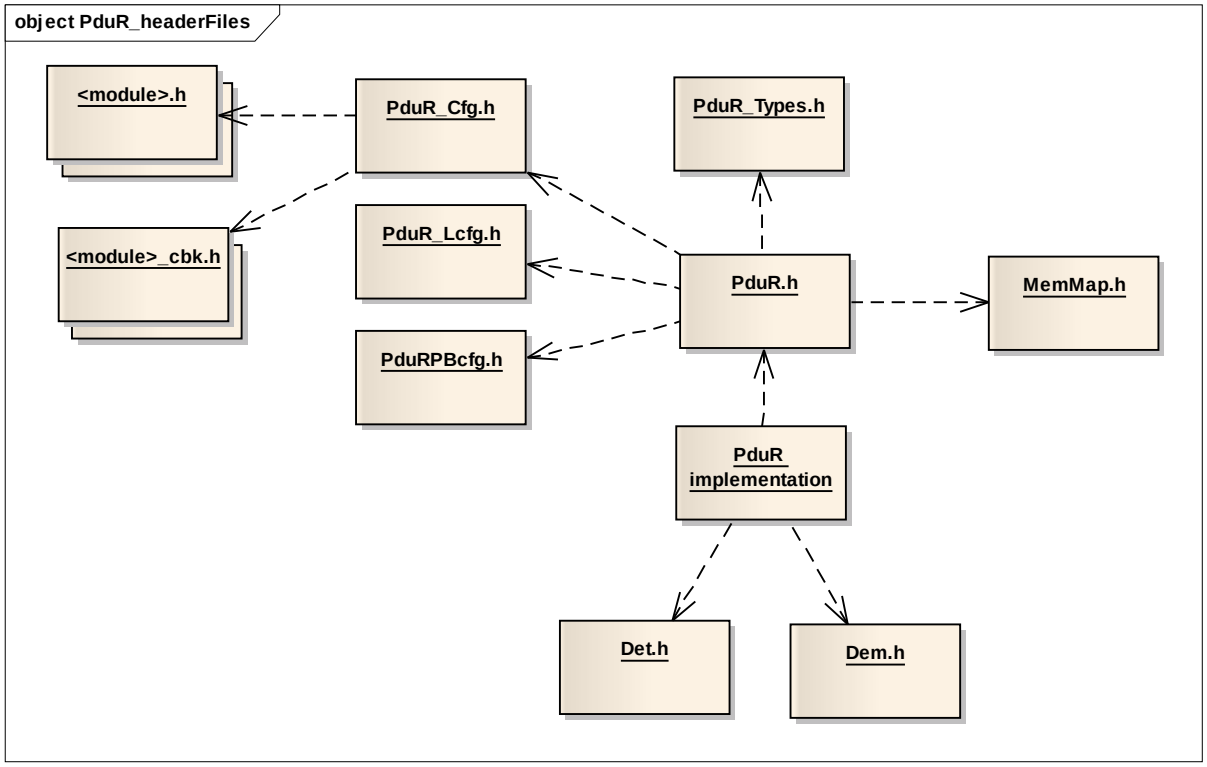

Figure 3: File Structure

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.

5.2 Version check

[PDUR0774] ⌈The PDU router module shall perform Inter Module Checks to avoid integration of incompatible files. The imported included files shall be checked by preprocessing directives. The following version numbers shall be verified (see **PDUR0778**) :

- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the PduR module. If the values are not identical to the expected values, a compile error shall be reported. ⌋ (BSW004)

6 Requirements traceability

Document: General Requirements on Basic Software Modules [3]

Requirement	Satisfied by
-	PDUR0667
-	PDUR0732
-	PDUR164
-	PDUR488
-	PDUR381
-	PDUR299
-	PDUR0779
-	PDUR435
-	PDUR375
-	PDUR0771
-	PDUR233
-	PDUR429
-	PDUR433
-	PDUR0670
-	PDUR333
-	PDUR0689
-	PDUR0781
-	PDUR518
-	PDUR551
-	PDUR232
-	PDUR0784
-	PDUR632
-	PDUR633
-	PDUR0745
-	PDUR0731

-	PDUR0715
-	PDUR287
-	PDUR101
-	PDUR643
-	PDUR0675
-	PDUR285
-	PDUR432
-	PDUR0727
-	PDUR0780
-	PDUR256
-	PDUR623
-	PDUR490
-	PDUR0740
-	PDUR0697
-	PDUR626
-	PDUR227
-	PDUR0687
-	PDUR649
-	PDUR319
-	PDUR286
-	PDUR0788
-	PDUR638
-	PDUR0783
-	PDUR430
-	PDUR662
-	PDUR0690
-	PDUR428
-	PDUR362
-	PDUR0766

-	PDUR306
-	PDUR231
-	PDUR436
-	PDUR161
-	PDUR303
-	PDUR627
-	PDUR621
-	PDUR629
-	PDUR280
-	PDUR0708
-	PDUR0736
-	PDUR0747
-	PDUR406
-	PDUR487
-	PDUR622
-	PDUR0765
-	PDUR100
-	PDUR504
-	PDUR0710
-	PDUR0683
-	PDUR648
-	PDUR301
-	PDUR624
-	PDUR549
-	PDUR324
-	PDUR218
-	PDUR0716
-	PDUR0767
-	PDUR0726

-	PDUR0665
-	PDUR654
-	PDUR0696
-	PDUR0714
-	PDUR0776
-	PDUR0707
-	PDUR635
-	PDUR424
-	PDUR217
-	PDUR645
-	PDUR619
-	PDUR0744
-	PDUR657
-	PDUR0773
-	PDUR293
-	PDUR0785
-	PDUR0746
-	PDUR0782
-	PDUR329
-	PDUR630
-	PDUR663
-	PDUR0733
-	PDUR482
-	PDUR331
-	PDUR327
-	PDUR326
-	PDUR661
-	PDUR307
-	PDUR0749

-	PDUR241
-	PDUR0772
-	PDUR489
-	PDUR0669
-	PDUR0676
-	PDUR0718
-	PDUR369
-	PDUR0787
-	PDUR365
-	PDUR0717
-	PDUR618
-	PDUR330
-	PDUR0673
-	PDUR0666
-	PDUR644
-	PDUR625
-	PDUR0725
-	PDUR507
-	PDUR216
-	PDUR325
-	PDUR0734
-	PDUR317
-	PDUR634
-	PDUR255
-	PDUR103
-	PDUR292
-	PDUR106
-	PDUR0709
-	PDUR308

-	PDUR0705
-	PDUR423
-	PDUR0769
-	PDUR0786
-	PDUR328
-	PDUR647
-	PDUR512
-	PDUR332
-	PDUR631
-	PDUR207
-	PDUR234
-	PDUR104
-	PDUR0661
-	PDUR434
-	PDUR640
-	PDUR637
-	PDUR646
BSW003	PDUR0762
BSW00325	PDUR0777
BSW00326	PDUR0777
BSW00334	PDUR0777
BSW00335	PDUR0777
BSW00336	PDUR0777
BSW00341	PDUR0777
BSW00343	PDUR0777
BSW00345	PDUR226, PDUR0748
BSW00347	PDUR0777
BSW00348	PDUR0777
BSW00353	PDUR0777

BSW00357	PDUR0777
BSW00358	PDUR334
BSW00361	PDUR0777
BSW00373	PDUR0777
BSW00375	PDUR0777
BSW00376	PDUR0777
BSW00379	PDUR0762
BSW00380	PDUR226
BSW00381	PDUR226
BSW00386	PDUR0777
BSW00398	PDUR0777
BSW004	PDUR0774
BSW00400	PDUR0743
BSW00406	PDUR119
BSW00407	PDUR338
BSW00409	PDUR0777
BSW00411	PDUR338
BSW00412	PDUR0748
BSW00413	PDUR0777
BSW00414	PDUR334
BSW00415	PDUR132
BSW00416	PDUR0777
BSW00417	PDUR0777
BSW00419	PDUR226
BSW00422	PDUR0777
BSW00423	PDUR0777
BSW00424	PDUR0777
BSW00425	PDUR0777
BSW00428	PDUR0777

BSW00432	PDUR0777
BSW00433	PDUR0777
BSW00437	PDUR0777
BSW00438	PDUR0743
BSW00439	PDUR0777
BSW00441	PDUR0742
BSW00443	PDUR0777
BSW00444	PDUR0777
BSW00445	PDUR0777
BSW00446	PDUR0777
BSW00447	PDUR0777
BSW00449	PDUR0777
BSW00450	PDUR0777
BSW005	PDUR0777
BSW06003	PDUR162
BSW06012	PDUR437
BSW06020	PDUR0764
BSW06049	PDUR160
BSW06055	PDUR0777
BSW06056	PDUR0777
BSW06061	PDUR0777
BSW06064	PDUR0777
BSW06077	PDUR0777
BSW06089	PDUR0777
BSW06097	PDUR341
BSW06098	PDUR0777
BSW06099	PDUR0777
BSW06103	PDUR221
BSW06119	PDUR589

BSW06120	PDUR615, PDUR617
BSW101	PDUR334
BSW160	PDUR0777
BSW161	PDUR0777
BSW162	PDUR0777
BSW164	PDUR0777
BSW168	PDUR0777
BSW170	PDUR0777
BSW172	PDUR0777

7 Functional Specification

The PDU Router module is an I-PDU transfer unit placed above interface modules and transport protocol modules (lower layer modules) and below COM and DCM (upper layer modules), see Figure 2.

Beside the PDU Router module is the I-PDU Multiplexer (IpduM) module [8] that 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 to local module(s):** receive I-PDUs from lower layer modules and forward them to upper layer modules,
- **PDU Transmission from local module(s):** transmit I-PDUs to lower layer modules on request of upper layer modules,
- **PDU Gateway:**
  (1) receive I-PDUs from an interface module and transmit the I-PDUs immediately via the same or other communication interface module(s)
  (2) receive I-PDUs from a transport protocol module and transmit the I-PDUs via the same or other transport protocol module(s).

The combination PDU Reception and PDU Gateway is allowed. Example: The COM module is receiving an I-PDU in the same time that it is gatewayed to another lower layer module.

7.1 I-PDU handling

[PDUR160] ⌈The PDU Router module shall transfer an I-PDU without modification in a consistent manner from the source module to the destination module(s). ⌋

(BSW06049)

An I-PDU is identified by the I-PDU ID and/or the symbolic name (i.e. the SymbolicNameValue of the container of the I-PDU) [7]. For debugging and post-build the I-PDU ID is required because the I-PDU must be identified after the PDU Router module is compiled. If the PDU router module is pre-compile (i.e. in source code) the symbolic names may be used, see Specification of ECU Configuration [7].

Each BSW module that handles I-PDUs and provides an API for I-PDUs must contain a list of I-PDU IDs [7]. This means that each called module will have a look-up table identifying the I-PDU.

Example: The COM module calls PduR_ComTransmit (here the PDU Router module will list the I-PDU ID), the PDU Router module will call CanIf_Transmit (here the CanIf module configuration will list the I-PDU ID), the CanIf will call PduR_CanIfTxConfirmation (here the PDU Router module configuration will list the I-PDU ID), and PDU Router module will call Com_TxConfirmation (here the COM module configuration will list the I-PDU ID). The example is illustrated in the following Figure 4 (only I-PDU ID is shown as parameter):

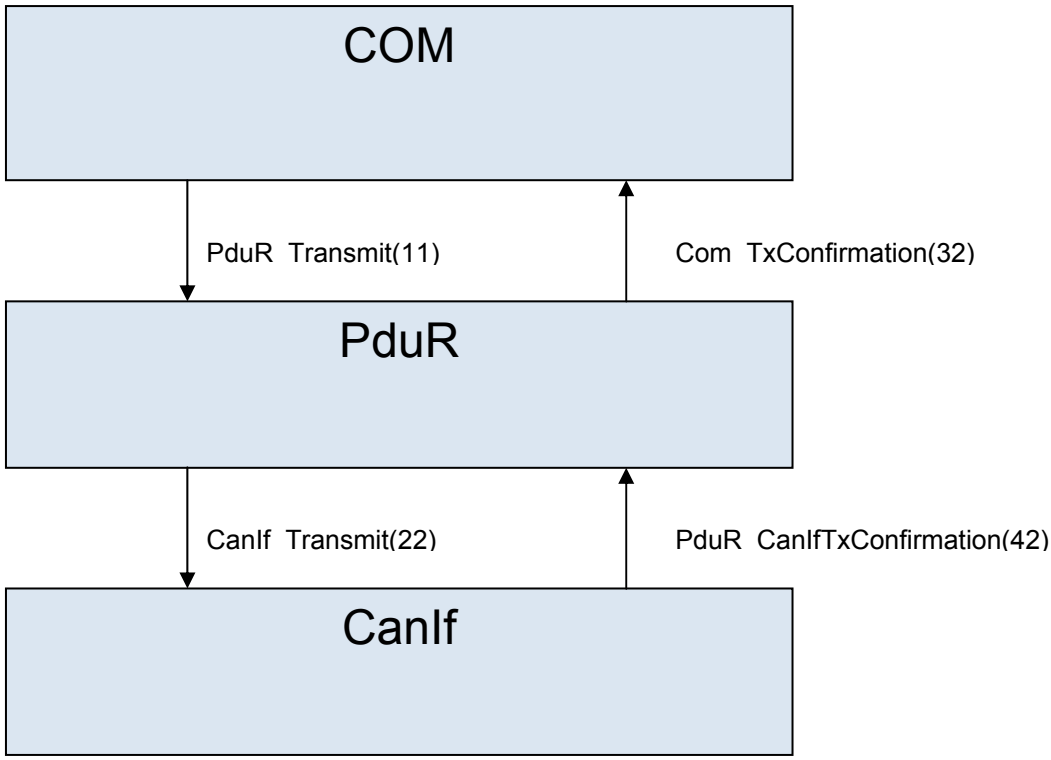

Figure 4 - I-PDU ID example

[PDUR161] ⌈The PDU Router module shall identify routing path uniquely by the combination of source module I-PDU ID (located in the PDU Router configuration) and destination I-PDU IDs (located in the called destination module configurations).⌋ ( )

[PDUR0766] ⌈The PDU Router module shall convert the I-PDU ID to the destination module(s) for both transmission path and confirmation/indication path. ⌋ ( )

Example: The COM module transmits an I-PDU to CanIf and LinIf. The PduR_ComTransmit is called. The PDU Router module will convert the source I-PDU ID (PDU Router module configuration) to one I-PDU ID for LinIf (LinIf module configuration) and one I-PDU ID for CanIf (CanIf module configuration). The PduInfoType value received from the COM module is copied to the CanIf and LinIf modules without change.

Example: When the LinIf have successfully transmitted an L-PDU it will call PduR_LinIfTxConfirmation with a I-PDU ID, then the PDU Router module will convert this I-PDU ID and forward the call to COM using Com_TxConfirmation with the converted I-PDU ID.

[PDUR162] ⌈The PDU Router module shall only route I-PDUs according to the static routing paths in the configuration. No dynamical routing shall be used. ⌋(BSW06003)

[PDUR618] ⌈In case an I-PDU is multicasted or gatewayed to more than one destination, the configuration parameter Routing path's HandleId denotes the order in which the destinations are served. The lowest HandleId will be served first. ⌋ ( )

Note: minimum routing routing tables will always be prioritized, see 7.6.

7.1.1 I-PDU Reception to upper module(s)

The receive operation of the PDU Router module is always be finalized by an indication (PduR_<Lo>RxIndication or PduR_<LoTp>RxIndication) from a lower layer module (communication interface module or 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.

7.1.1.1 Communication Interface

The source communication interface module indicates a received I-PDU by calling PduR_<Lo>RxIndication. The I-PDU may have multiple destination local modules as configured by the routing path.

[PDUR164] ⌈The PDU Router module shall provide 1:n routing for an I-PDU received from a communication interface module and routed to upper layer module(s).

Example: An I-PDU is received on CanIf and forwarded to COM.

Note: An I-PDU may be received by one or more upper modules in the same time as gatewayed to one or more communication interfaces, see 7.1.3. ⌋( )

[PDUR621] ⌈When the PduR_<Lo>RxIndication is called the PDU Router module shall call <Up>_RxIndication for each destination upper module. ⌋( )

[PDUR0744] ⌈If the I-PDU is received by a local module the PDU Router shall not check the length of the I-PDU, just forward the indication to the upper layer module.

Since the PDU Router module will not buffer this I-PDU it does not have to reject I-PDU that are longer/shorter than configured. ⌋( )

7.1.1.2 Transport Protocol

In case of the transport protocol module the PDU Router module is first notified with a start of reception notification when receiving a first frame (FF) or single frame (SF). This call is be forwarded to the related upper layer module by calling <Up>_StartOfReception. The payload of each segment (N-PDU) is to be copied in the destination upper layer module by the within subsequent <Lo>_CopyRxData calls. 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 will forward this indication to the related upper layer module by calling <Up>_TpRxIndication.

Reception of I-PDU through Transport Protocol module may have only one upper layer module configured by the routing paths.

[PDUR0673] ⌈The PDU Router module shall provide 1:1 routing for an I-PDU received from a source transport protocol module and routed to one destination upper module.

Example: A functional addressed request (in a SF) is received from the CanTp module and routed to the DCM module. ⌋( )

[PDUR549] ⌈When a source transport protocol module indicates the start of a reception using PduR_<Lo>StartOfReception, the PDU Router module shall forward the request to the destination upper layer module by calling <Up>_StartOfReception. ⌋( )

[PDUR0749] ⌈If the upper layer returns with an error in the <Up>_StartOfReception the PDU Router shall not expect a PduR_<LoTp>RxIndication from the lower layer transport protocol module. ⌋( )

[PDUR623] ⌈The PDU Router shall forward the return value of the <Up>_StartOfReception to the source transport protocol. ⌋( )

[PDUR428] ⌈When a source transport protocol module requests the PDU Router module to copy the received data using PduR_<LoTp>CopyRxData, the PDU Router module forward the request to the destination upper layer module by calling <Up>_CopyRxData. ⌋( )

[PDUR429] ⌈When a source transport protocol module calls PduR_<LoTp>RxIndication indicating reception of the complete I-PDU, the PDU Router module shall forward the indication to the destination upper layer module by calling <Up>_TpRxIndication. ⌋( )

[PDUR207] ⌈If the source transport protocol module reports an error using PduR_<LoTp>RxIndication, the PDU Router module shall not perform any error handling other than forwarding the indication to the upper layer module. ⌋( )

The PDU Router assumes the Transport Protocol behaves correctly therefore no

explicit error handling is needed for production code. However it may be helpful to have errors reported to DET in certain cases:

[PDUR624] ⌈If the PduR_<Lo>StartOfReception is called with an I-PDU ID that is already in process, the PDU Router module shall forward the call to the upper module and report PDUR_E_DUPLICATE_IPDU_ID to DET, when PduRDevErrorDetect is enabled.⌋( )

7.1.2 I-PDU Transmission from upper module(s)

The transmit operations of the destination 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 destination lower layer module. If the PDU Router module is notified by destination lower layer modules via PduR_<Lo>TxConfirmation (communication interface) or PduR_<LoTp>TxConfirmation (transport protocol) after the I-PDU has been transmitted the PDU Router module will forward this indication to the upper layer module via <Up>_TxConfirmation (communication interface) or <Up>_TpTxConfirmation (transport protocol).

The transmit operation of the PDU Router module is triggered by a PDU transmit request from a source upper layer source module and forwards the request to a destination lower layer module.

[PDUR629] ⌈The I-PDU shall not be buffered in the PDU Router module in case of PDU transmission from a source upper layer module.⌋( )

7.1.2.1 Multicast

The multicast feature is separated to an own section since there are issues using this feature as described in section 4.1.1 and also below.

The PDU router is capable of multicast (1:n;n>1) an I-PDU transmitted by a local module (e.g. COM module) to one or more communication interface modules or transport protocol modules. This introduces some limitations since the local module is not aware to how many destinations the I-PDU will be transmitted to.

Multicast of I-PDU has the following restrictions (note that these are not requirements since the features are in other documents):

- No update-bits can be used
- No sequence counters can be used
- If deadline monitoring is used then all destinations must provide transmit confirmations

Further requirements that are directly handled by the PDU Router module:

[PDUR218] ⌈If the provided I-PDU ID 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.

Note that communication interfaces returning with E_OK will transmit their data either directly or via trigger transmit. ⌋( )

[PDUR633] ⌈If there are more than one lower layer destination modules in a transmission request (1:n, n>1), all of these modules must either be communication interface modules or transport protocol modules. Not a mix of them. ⌋( )

Example: Above requirement means basically that the COM module cannot request a transmission, and then the PDU Router module routes the transmission to CanTp module and CanIf module using the same I-PDU.

[PDUR635] ⌈If there is a multicast (1:n, n>1) transmission the PDU Router module shall call them in routing-path ID order with lowest ID first. ⌋( )

To allow upper layer modules to have timeout monitoring and release locked buffers the following apply:

[PDUR589] ⌈In case of a multicast (1:n, n>1), communication interface transmission, the PDU Router shall forward the transmit confirmation to the upper layer module for the last I-PDU transmitted, i.e. the last transmission confirmation received from the communication interface module. ⌋(BSW06119)

Note: Above requirement can only work if the all the destinations will provide with tx confirmations.

Implementation note: When the source module requests a transmission and the PduR will make a multicast (1:n, n>1), all the I-PDUs in the request and the multicast will have different I-PDU IDs. Therefore the PduR must remember the I-PDU ID from the transmission request so the transmission can be confirmed correctly.

7.1.2.2 Communication Interface

There are three ways that I-PDUs can be transmitted on the communication interface:

1. Direct data provision - where the upper module is calling the PduR_<Up>Transmit function, the PDU Router module forwards the call to <Lo>_Transmit and the data is copied by the lower communication interface module in the call.

2. Trigger transmit data provision – where the lower communication interface module requests transmission of an I-PDU by using the PduR_<Lo>TriggerTransmit, and PDU Router module forwards the call to PduR_<Up>TriggerTransmit and the data is copied by the upper module.

3. Where the upper module is calling the PduR_<Up>Transmit function, the PDU Router module forwards the call to PduR_<Lo>Transmit and the data is not copied by the lower module (communication interface module). The data will later be requested by the lower layer using PduR_<Lo>TriggerTransmit.

The confirmation of the transmission of the I-PDU is the same for the direct and trigger transmit data provisions:

[PDUR627] ⌈When the communication interface module calls PduR_<Lo>TxConfirmation the PDU Router shall call <Up>_TxConfirmation in the upper module.⌋( )

[PDUR0745] ⌈If the I-PDU is transmitted by an upper layer module the PDU Router module shall not check the length of the I-PDU.⌋( )

[PDUR625] ⌈When source upper layer module calls PduR_<Up>Transmit the PDU Router shall call <Lo>_Transmit for each destination communication interface module.⌋( )

[PDUR626] ⌈If singlecast (1:1) the return value of the <Lo>_Transmit call shall be forwarded to the source upper layer module.⌋( )

Trigger transmit data provision

The upper layer module must be informed if it has to reset the update-bits or not and handle the I-PDU counter in a proper way.

[PDUR430] ⌈The PDU Router module shall forward a PduR_<Lo>IfTriggerTransmit request by the communication interface module to the upper module by calling <Up>_TriggerTransmit.⌋( )

[PDUR661] ⌈The PDU Router module shall copy the return value from the <Up>_TriggerTransmit to the lower layer module.⌋( )

[PDUR0773] ⌈After calling <Lo>_Transmit and the destination communication interface is using trigger transmit data provision the PDU router shall return E_OK if all communication interface modules return E_OK.⌋( )

Multicast

Following requirements apply if the I-PDU is multicasted 1:n; n>1:

[PDUR0675] ⌈If multicast (1:n, n>1) then E_OK shall be returned if all destination communication interfaces return E_OK (pessimistic approach).⌋( )

[PDUR0718] ⌈In case of a multicast transmission request and at least one destination communication interface module returns E_NOT_OK after calling <Lo>_Transmit the PDU Router shall return E_NOT_OK.⌋( )

[PDUR0772] ⌈After calling <Lo>_Transmit and in case of multicast and at least one destination communication interface is using trigger transmit data provision the PDU router shall return E_OK if all communication interface modules return E_OK.⌋( )

Error handling

For errors occurred using singlecast or multicast over communication interface modules, no specific error handling is done. Errors in return values are forwarded to the source upper layer module.

7.1.2.3 Transport Protocol

Transmitting I-PDU using transport protocol has two flavors, singlecast and multicast. A singlecast (1:1) transmission consists of one source upper layer module and one destination lower layer transport protocol module. A multicast (1:n, n>1) transmission consists of more than one destination lower layer transport protocol module. The PDU Router module will not check if the transmission request contains a single N-PDU (SF) or multiple N-PDU (FF, CF, …), though multicast transport prototcol transmissions are restricted to SF transmissions.

Initiation of transmission of I-PDU is made by a PduR_<Up>Transmit request by an upper layer source module. The PduR will forward the request to one or more destination lower layer transport protocol modules using <Lo>_Transmit according to the routing paths. Note that the <Lo>_Transmit may or may not contain data.

The destination module(s) will request data by calling the PduR_<LoTp>CopyTxData. Retransmission (if supported by the transport protocol) of data is made by the RetryInfoType parameter. Finalizing the transmission the destination module(s) calls the PduR_<LoTp>TxConfirmation, which is forwarded to the source upper layer module.

[PDUR634] ⌈When an upper layer module calls the PduR_<Up>Transmit the PDU Router module shall call <LoTp>_Transmit for each destination transport protocol module.⌋( )

[PDUR299] ⌈When a destination transport protocol module calls PduR_<LoTp>CopyTxData the PDU Router module shall call <Up>_CopyTxData in the source upper layer module.⌋( )

[PDUR0676] ⌈The return value from the <Up>_CopyTxData shall be forwarded to

the calling destination lower layer transport protocol module. ⌋( )

[PDUR301] ⌈In case of singlecast the PDU Router module shall forward the confirmation PduR_<LoTp>TxConfirmation from the lower layer transport protocol module to upper layer module using <Up>_TpTxConfirmation. ⌋( )

[PDUR432] ⌈In case of singlecast and after calling <Lo>_Transmit then the PDU Router module shall return with the same return value to the calling PduR_<Up>Transmit from source upper layer module. ⌋( )

[PDUR435] ⌈In case a transport protocol module reports PduR_<LoTp>TxConfirmation with result other than NTFRSLT_OK the PDU Router shall forward the result in the <Up>_TpTxConfirmation to the source upper layer module. Further calls from transport protocol module(s) calling PduR_<LoTp>CopyTxData the PDU Router module shall return directly with BUFREQ_E_NOT_OK. ⌋( )

Multicast transmission

This subsection contains specific requirements for the multicast transmission of I-PDU using transport protocol modules.

Since the 1:n, n>1 routing is made in the PDU Router module the PDUR Router module must request the same data several times from the source upper layer module. Also the confirmation of the multicast must be handled specifically.

As the upper layer shall copy the same data several times, the PDU Router will use the RetryInfoPtr [4] in order to query the same data several times. The RetryInfoPtr contains a state type called TpDataState.

[PDUR631] ⌈The first request of PduR_<LoTP>CopyTxData in a multicast session shall be forwarded with TpDataState set to TP_CONFPENDING. ⌋( )

[PDUR632] ⌈All following calls of PduR_<LoTp>CopyTxData requests shall be forwarded with TP_DATARETRY to allow the same data to be copied.

After all transport protocols have received their data the PDU Router module may

confirm the data to the upper layer module.⌋( )

[PDUR0765] ⌈In case of multicast and when receiving the last confirmation PduR_<LoTp>TxConfirmation from the lower layer transport protocol module, the PDU Router module shall call upper layer module using <Up>_TpTxConfirmation.⌋( )

[PDUR433] ⌈In case of a multicast transmission request and at least one destination transport protocol module returns E_NOT_OK after calling <LoTp>_Transmit then the PDU Router module shall return with the same code to the calling source upper layer module.⌋( )

The other transport protocol modules may return E_OK, and therefore these modules will call the PduR_<LoTp>CopyTxData. Since the source upper layer module has been informed the transmission was a failure the other transport protocol modules should stop as soon as possible.

[PDUR434] ⌈In case of a multicast transmission request and at least one destination transport protocol module returns E_NOT_OK after calling <LoTp>_Transmit the PDU Router shall return BUFREQ_E_NOT_OK if other modules are calling PduR_<Lo>CopyTxData.⌋( )

Error handling

[PDUR0779] ⌈If PduR_<LoTP>CopyTxData is called with TpDataState TP_DATARETRY and TxTpDataCnt ≠ 0 in case of a multicast transmission request, the PDU Router shall raise the development error `PDUR_E_PARAM_INVALID`.⌋( )

The PDU Router module will not take specific actions on errors occurred, the errors will be forwarded to the source upper layer module. Appropriate error handling is in the responsibility of the upper layer module.

7.1.3 I-PDU Gateway

The PDU Router module supports gatewaying of I-PDUs from one source bus to one or more destination busses. The difference from a transmission and reception from/to a local module is that the PDU Router module must be a receiver and transmitter at

the same time, and in some cases also provides buffering for the I-PDU.

The gateway requirements are deliberately separated to allow an efficient implementation of the PDU Router module in case gatewaying is not needed. In case the PDU Router module allows gatewaying of I-PDUs, these requirements are seen as additional and not replacing previous requirements.

Following list gives an overview of the features of the I-PDU gateway:

- I-PDUs may be gatewayed from a source communication interface module to one (1:1) or more destination communication interface modules (1:n I-PDU gateway)
  - For each destination the PDU Router module may buffer each destination of an I-PDU in configurable depth (i.e. FIFO if more than one I-PDU).
  - An I-PDU may be received by an upper layer module in the same time as gatewayed to n destination communication interface(s).
- I-PDUs transported using TP may be gatewayed to one or more destination TP modules, with following scope:
  - Only Single Frames (SF) may be gatewayed to more than one destination TP module or local module (e.g. DCM).
  - I-PDU transported in multiple N-PDUs may be gatewayed “on-the-fly” to one destination, meaning that complete I-PDU does not need to be received before starting transmission on the destination TP module
  - I-PDU transported in multiple N-PDUs may be gatewayed to another TP module or received by a local module, not both.
  - I-PDUs transported using TP module may not be FIFO buffered, i.e. the PDU Router may not buffer more than one I-PDU.
- I-PDUs can only be gatewayed between communication interface modules or TP modules, not a mix of them. For example an I-PDU cannot be received from CanIf and gatewayed to LinTp.

This 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 I-PDU ID.
Note that in this section “Src” and “Dst” are used for the configurable APIs. This is just to be clear which call belongs to the source module and destination module.

[PDUR638] ⌈An I-PDU may only be gatewayed between communication interfaces or TPs, not a mix of them.

Example: An I-PDU received from FrIf may not be gatewayed to CanTp. ⌋( )

7.1.3.1 Communication interface modules

An I-PDU can be configured to be received on one communication interface module and gatewayed to n communication interface modules including local module(s), i.e. 1:n gatewaying. For gatewaying it is also possible to configure a FIFO for each destination communication interface module (not local module however).

[PDUR436] ⌈The PDU Router module shall support routing of I-PDUs between a source communication interface module and one or more destination communication interface modules (1:n gatewaying). ⌋( )

[PDUR437] ⌈ The PDU Router module shall support routing of I-PDUs between communication interface modules with immediate transmission (without rate generation by PDU Router) ⌋(BSW06012)

Routing of I-PDUs between communication interface modules with different period or rate (rate conversion) is not supported, this can be done via the COM module using signal gateway. In this case the I-PDU has to be routed to the COM module.

There are two flavors of gatewaying an I-PDU depending on the the destination interface module. The used flavor is controlled by the configuration:

- **[PDUR303]** ⌈Direct data provision: The PduRDestPduDataProvision of the destination I-PDU is configured to PDUR_DIRECT. When <DstLo>_Transmit is called the <DstLo> module copies the data and the PDU Router does not buffer the transmitted I-PDU any longer. ⌋( )

- **[PDUR306]** ⌈Trigger transmit data provision: The PduRDestPduDataProvision of the destination I-PDU is configured to PDUR_ TRIGGERTRANSMIT. When <DstLo>_Transmit is called the <DstLo> module does not copy the data and the PDU Router module shall buffer the I-PDU and wait for the PduR_<DstLo>TriggerTransmit call from the <DstLo> module. ⌋( )

[PDUR0661] ⌈The PDU Router shall buffer the latest I-PDU in case of a trigger transmit data provision.⌋( )

The reason why it must be stored for trigger transmit data provision is that the destination communication interface may transmit the I-PDU according to a schedule. Then the communication interface will call the PduR_<DstLo>TriggerTransmit without a preceding <DstLo>_Transmit call.

[PDUR662] ⌈If the destination communication interface module is requesting the I-PDU buffer using PduR_<DstLo>TriggerTransmit and the buffer is not available the return value E_NOT_OK shall be used.⌋( )

Note that for a gateway of an I-PDU the PduR_<Lo>TxConfirmation is not interesting (except for FIFO of a direct data provision Pdu).

[PDUR640] ⌈If the destination communication interface module confirms the transmission of the I-PDU using PduR_<DstLo>TxConfirmation and destination is not a direct data provision Pdu with FIFO buffer the PDU Router module shall not do anything.⌋( )

Note: It is not required to have one buffer for each destination. In following cases sharing the same buffer may be meanigfull to save RAM:

- In case of multicast and buffering is needed for trigger transmit data provision (This optimization may not be possible when FIFO buffering is used)
- In case never more then one routing path to one destination Pdu is switched on at the same time (controlled by routing path groups)
- In case the user has the knowledge that at runtime never more then one routing path to one destination Pdu is triggered by a reception of a source Pdu

It is possible that an I-PDU that will be gatewayed will have different lengths. Therefore:

[PDUR0783] ⌈In case the I-PDU is gatewayed without buffering in the PDU Router, the PDU Router shall forward the length without checking, just calling the transmit function(s) of the destination modules.⌋( )

[PDUR0746] ⌈In case the I-PDU is buffered in the PDU Router module: The PDU Router module shall copy the data of of the I-PDU up to smallest of the following values:

- the received data length (PduLength of received I-Pdu)
- the configured maximum PduLength in the buffer (PduRPduMaxLength). In this case the rest of the received I-PDU shall be dropped. ⌋( )

[PDUR0784] ⌈When the I-PDU is transmited (direct data provision) or copied (trigger transmit data provision) from the PduR buffer to the destination module the PduR shall use the number of bytes which was copied to the buffer as Pdu length.⌋( )

Note: [PDUR0746] give the user the possibility to avoid buffer over run when PduR_<Lo>TriggerTransmit is called. The user needs to configure PduRMaxPduLength not greater then the length of the destination I-Pdu.

FIFO

[PDUR0785] ⌈If PduRTxBufferDepth is configured to a value greater then 1 the Tx Pdu buffer shall have a first in – first out (FIFO) behavior. ⌋( )

In the following chapter the term “FIFO” or “FIFO queue” is used as a synomym for the Tx I-Pdu buffer of the PduR.

The FIFO has states and these states may change when calling various PduR API’s called from different contexes. E.g. a PduR_<SrcLo>RxIndication call could be interrupted by a PduR_<DstLo>TxConfirmation call. Thus there is a need to protect those concurrent calls.

If a FIFO is used in case of direct data provision the destination I-PDU must be configured to call the PduR_<DstLo>TxConfirmation, see PduRTransmissionConfirmation.

[PDUR307] ⌈It shall be possible to configure a FIFO for each destination I-PDU⌋( )

[PDUR0665] ⌈When PduR_<SrcLo>RxIndication is called and the FIFO queue is empty in case of direct data provision <DstLo>_Transmit shall be called directly. The FIFO stays empty. ⌋( )

[PDUR0786] ⌈When PduR_<SrcLo>RxIndication is called and the FIFO queue is empty in case of trigger transmit data provision the received I-PDU shall be copied to the FIFO and <DstLo>_Transmit shall be called.⌋( )

[PDUR0787] ⌈When PduR_<SrcLo>RxIndication is called and the FIFO queue is not empty then the received I-PDU shall be copied as latest entry.⌋( )

[PDUR0666] ⌈When PduR_<DstLo>TriggerTransmit is called the oldest FIFO entry shall be copied and then removed. If afterwards the FIFO queue is not empty <DstLo>_Transmit shall be called with the oldest I-PDU of the FIFO.⌋( )

Note: In case of the destination module is FrIf the FrIfCounterLimit of the Pdu needs to be configured > 1 because the new transmit will be called before the counter is decremented. For LinIf there is no such a constraint, however FIFO queue routing to sporadic frames is not supported.

[PDUR0667] ⌈When PduR_<DstLo>TxConfirmation is called and the FIFO queue is not empty in case of direct data provision <DstLo>_Transmit shall be called with the oldest I-PDU of the FIFO. The transmitted I-PDU shall be removed afterwards.⌋( )

Error handling

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.

[PDUR256] ⌈If the PDU Router module buffers the I-PDU and destination communication interface module returns E_NOT_OK after calling <DstLo>_Transmit, the PDU Router shall discard this I-PDU.⌋( )

Here the destination returned E_NOT_OK for some reason, will also report this error. The PDU Router module cannot do anything else than discarding the I-PDU.

[PDUR0788] ⌈If <DstLo>_Transmit(), called with an I-PDU from the FIFO buffer, returns E_NOT_OK the I-PDU shall be removed from the FIFO and the next FIFO entry shall be transmitted, if available.

[PDUR255] ⌈If the FIFO is full and a new PduR_<SrcLo>RxIndication is called, the FIFO shall be flushed⌋( )
Note: That means in case of PduRTxBufferDepth is configured to 1 and the PduRDestPduDataProvision is configured to PDUR_TRIGGERTRANSMIT the new I-Pdu will be always copied within the next PduR_<Lo>TriggerTransmit call. That is a "Last is best" behaviour.

[PDUR0670] ⌈If the FIFO is flushed the PDU Router shall report PDUR_E_PDU_INSTANCES_LOST to the DET module.⌋( )

[PDUR0669] ⌈If the FIFO is flushed the new I-PDU delivered by the PduR_<SrcLo>RxIndication shall be handled as if the FIFO is empty.⌋( )

The new I-PDU that causes the FIFO flush will be served and not discarded.

[PDUR319] ⌈In case of the destination I-PDU is configured to PDUR_ TRIGGERTRANSMIT but no buffer is configured for the routing path the PDU Router module shall not call <DstLo>_Transmit for this routing path.⌋( )

Above requirement shows that the PDU Router module is gatewaying in a best effort manner. If resources are not available then the gateway operation stops as soon as possible.

7.1.3.2 Transport Protocol

Gatewaying I-PDU from a source transport protocol to one or more destination transport protocol module may either be gatewayed direct as a complete I-PDU (complete set of N-PDUs building up the I-PDU is received before transmitted) or as fragmented I-PDU (routing on-the-fly) where a configured number of bytes (threshold) are received before transmission.

The PDU Router will gateway the payload only, and will not be aware of transport protocol details such as SF, FF, CF, PCI etc.

On a transport protocol module an I-PDU can be transported in multiple N-PDUs (FF and CFs) or in a single N-PDU (SF). The typical case is that an I-PDU transported in multiple N-PDUs are not multicast (i.e. physical addressing) and in single N-PDU

may be multicast (i.e. functional addressing). This is however no restriction on the PDU router since it only gateways I-PDUs (and not N-PDUs).

It is not possible for the PDU Router module to check if the I-PDU received from the source transport protocol is really contained in a SF and if the destination transport protocol module really can be contained in a SF. It is up to the system designer to assure that the transport protocols behaves properly.

For example: A SF received on CAN and shall be transmitted on two LIN busses. The received SF can carry up to data 6 bytes but a SF on LIN only up to data 5 bytes. Therefore the SF on CAN is limited to data 5 bytes if gatewayed to the two LIN busses.

Note that an I-PDU transported over transport protocol modules may also be gatewayed frame by frame directly through the communication interfaces (i.e. by gatewaying the N-PDUs directly). This requires no special treatment here of the PDU Router module and can be handled by gatewaying through communication interface modules, see 7.1.3.1. However, this requires that the source and destination busses have exactly same packing of N-PDUs (e.g. from CAN to CAN).

Direct gatewaying

[PDUR643] ⌈The PDU Router module shall receive complete I-PDU before gatewaying it to one or more destination transport protocol modules.⌋( )

[PDUR0683] ⌈If the I-PDU is gatewayed to one or more destination transport protocol modules, this I-PDU may be also received by one or more local upper layer modules, in this case 7.1.1.2 applies also.⌋( )

[PDUR551] ⌈The <DstLoTp>_Transmit shall be called on each destination transport protocol module within the PduR_<SrcLoTp>TpRxIndication, if result is NTFRSLT_OK.⌋( )

[PDUR0697] ⌈If PduR_<DstLoTp>CopyTxData is called and state is TP_CONFPENDING or TP_DATACONF then the PDU Router shall copy data equal to the PduInfoType length value or to the available buffer if lower than the PduInfoType length value.⌋( )

[PDUR0705] ⌈If PduR_<DstLoTp>CopyTxData is called and state is TP_DATARETRY then the PDU Router shall copy previous data according to TxTpDataCnt or to the available buffer if lower than the TxTpDataCnt.⌋( )

Gatewaying on-the-fly

In gatewaying on-the-fly the PDU Router module will start transmission to the destination transport protocol module when a specific threshold is reached.

[PDUR0708] ⌈Using gatewaying on-the-fly only one destination transport protocol module is allowed.⌋( )

[PDUR317] ⌈The PDU Router module shall start the TP transmission on the destination bus by calling <DstLoTp>_Transmit as soon as the Tx threshold has been reached for the specific destination.⌋( )

[PDUR0690] ⌈If gatewaying on-the-fly is used the PDU Router shall not support any retransmission. If the PDU Router receives a state other than TP_DATACONF the PDU Router shall immediately stop further processing of this I-PDU.⌋( )

[PDUR0707] ⌈If PduR_<DstLoTp>CopyTxData is called and state is TP_DATACONF then the PDU Router shall copy data equal to the PduInfoType length value or to the available buffer if lower than the PduInfoType length value.⌋( )

Common requirements

Following requirements applies to both direct gatewaying and routing-on-the-fly gatewaying.

[PDUR0696] ⌈If PduR_<DstLoTp>CopyTxData is called and state is TP_DATACONF then the PDU Router may free the already copied data.⌋( )

All destination transport protocol modules will confirm transmission of the I-PDU.

[PDUR637] ⌈When the PDU Router module receives the PduR_<DstLoTp>TxConfirmation, the PDU Router shall free the I-PDU buffer for this

destination. ⌋( )

[PDUR0740] ⌈If the transport protocol module calls PduR_<LoTp>CopyTxData or PduR_<LoTp>CopyRxData with length zero (PduInfoType.SduLength = 0) the PDU Router module shall return the size of the current available buffer or the current available data respectively. ⌋( )

Error handling

Error handling for I-PDUs gatewayed using transport protocol modules is simple, the PDU Router module will not do anything and rely that the transport protocol modules handles the communication errors properly.

[PDUR0687] ⌈If routing on-the-fly is used and PduR_<SrcLoTp>CopyRxData is called and the provided data cannot be stored in the buffer, then BUFREQ_E_NOT_OK shall be returned and the execution of the I-PDU gateway shall be stopped. ⌋( )

[PDUR0689] ⌈The PDU Router shall immediately stop further processing of the I-PDU if result value is not NTFRSLT_OK in the PduR_<SrcLoTp>RxIndication. ⌋( )

If the PDU Router cannot accept the I-PDU an shall be returned in the PduR_<LoTp>StartOfReception the PDU Router shall not expect a PduR_<LoTp>RxIndication from the lower layer transport protocol module.

7.2 Cancel transmission

An upper layer module may request cancellation of an I-PDU (transported by communication interface module or transport protocol module). The PDU Router module will forward the request to either one destination module (singlecast) or multiple destination modules (multicast).

The PduR_<Up>CancelTransmit is used to cancel communication interface I-PDU and to cancel transport protocol I-PDUs.

Cancel transmission will not be used for I-PDUs that are gatewayed from one bus to

another.

The cancel transmission is optional and enabled in the configuration per module, see PduRCancelTransmit configuration parameter.

7.2.1 Request

An upper layer module requests cancellation of an I-PDU, and the PDU router will forward the request to one or more destination modules according to the routing path.

[PDUR0725] ⌈If a cancel transmission is already ongoing for the same I-PDU, the old request shall be dropped and the new request shall be processed. ⌋( )

[PDUR0710] ⌈If the routing path for the requested I-PDUs is enabled, the I-PDU belongs to corresponding transport protocol or communication interface, and is not gatewayed, then PduR_<Up>CancelTransmit shall return E_OK. In all other cases E_NOT_OK shall be returned. ⌋( )

Following Table 1 describes the behavior in the PDU Router module when the PduR_<Up>CancelTransmit is called:

	Single destination	Multiple destinations
Communication interface module PduR_<Up>CancelTransmit	**[PDUR0721]** ⌈The PDU Router module shall call the <Lo>_CancelTransmit for the destination module of the I-PDU. ⌋( )	**[PDUR0723]** ⌈The PDU Router module shall call the <Lo>_CancelTransmit for each destination module of the I-PDU. ⌋( )
Transport protocol module PduR_<Up>CancelTransmit	**[PDUR0722]** ⌈The PDU Router module shall call the <LoTp>_CancelTransmit for the destination module of the I-PDU. ⌋( )	**[PDUR0724]** ⌈The PDU Router module shall call the <LoTp>_CancelTransmit for each destination module of the I-PDU. ⌋( )

Table 1 - PduR_<Up>CancelTransmit

Following Table 2 describes the behavior in the PDU Router module when the return value of <Lo>_CancelTransmit or <LoTp>_CancelTransmit is received:

	Single destination	Multiple destinations
Communication interface module <Lo>_CancelTransmit Transport protocol module <LoTp>_CancelTransmit	**[PDUR0700]** ⌈The PDU Router module shall return same return value to the calling upper layer module.⌋( )	**[PDUR0701]** ⌈E_OK shall be returned to the calling upper layer if all destination modules return E_OK, otherwise E_NOT_OK shall be returned.⌋( )

Table 2 - Return of PduR_<Up>CancelTransmit

7.2.2 Confirmation

The cancellation of an I-PDU will be only be confirmed by the destination transport protocol module(s) using PduR_<LoTp>TxConfirmation. The PDU Router module will forward the result to the calling upper layer module.

Following Table 3 describes the behavior in the PDU Router module when the PduR_<LoTp>TxConfirmation is called:

	Single destination	Multiple destinations
Transport protocol module	**[PDUR0728]** ⌈When PduR_<LoTp>TxConfirmation is called the PDU Router module shall forward using <Up>_TpTxConfirmation to the calling upper layer module⌋( )	**[PDUR0729]** ⌈When the last PduR_<LoTp>TxConfirmation is called the PDU Router module shall forward using <Up>_TpTxConfirmation to the calling upper layer module⌋( ) **[PDUR0730]** ⌈If all destination modules reports NTFRSLT_E_CANCELATION_OK then this value shall be forwarded upwards, otherwise NTFRSLT_E_CANCELATION_NOT_OK shall be forwarded⌋( )

Table 3 - PduR_<Lo/LoTp>TxConfirmation

7.3 Cancel reception

An upper layer module may request cancellation of an I-PDU transported on transport protocol module(s). The PDU router module will get a request through the PduR_<Up>CancelReceive. The confirmation of the cancellation request is made through the PduR_<LoTp>RxIndication and forwarded to <Up>_TpRxIndication using the results NTFRSLT_E_CANCELATION_OK respective NTFRSLT_E_CANCELATION_NOT_OK.

7.3.1 Request

[PDUR0731] ⌈If a receive cancellation is already ongoing for the same I-PDU, the old request shall be dropped and the new request shall be processed. ⌋( )

[PDUR0726] ⌈If the routing path for the requested I-PDUs is disabled, then PduR_<Up>CancelReceive shall return E_NOT_OK directly without any further action. ⌋( )

The flow of the I-PDU id for a received I-PDU is from lower to upper modules. Here it is used in the flow from upper to lower modules, since it is a received I-PDU.

[PDUR0736] ⌈The I-PDU id provided in the call is Rx I-PDU ID and therefore the PDU Router module shall be able to identify this I-PDU correctly. ⌋( )

[PDUR0727] ⌈When the PduR_<Up>CancelReceive is called the PDU Router module shall call the <LoTp>_CancelReceive for the destination transport protocol module of the I-PDU. ⌋( )

[PDUR0732] ⌈The return value of the <LoTp>_CancelReceive shall be forwarded to the upper layer module. ⌋( )

7.3.2 Indication

[PDUR657] ⌈When the destination transport protocol module calls PduR_<LoTp>RxIndication the PDU router module shall call the <Up>_RxIndication

in the calling upper layer module. ⌋( )

7.4 Change transport protocol parameter

It is possible for the upper layer modules to change a transport protocol parameter for a specific I-PDU. The upper layer will start with calling the PduR_<Up>ChangeParameter, the PDU Router module will forward to the correct transport protocol module using <LoTp>_ChangeParameter.

7.4.1 Request

[PDUR0733] ⌈When the PduR_<Up>ChangeParameter is called the PDU Router module shall call the <LoTp>_ChangeParameter for the destination transport protocol module of the I-PDU. ⌋( )

The flow of the I-PDU id for a received I-PDU is from lower to upper modules. Here it is used in the flow from upper to lower modules, since it is a received I-PDU.

[PDUR0747] ⌈The I-PDU id provided in the call is Rx I-PDU ID and therefore the PDU Router module shall be able to identify this I-PDU correctly. ⌋( )

[PDUR0734] ⌈The return value of the <LoTp>_ChangeParameter shall be forwarded to the upper layer module. ⌋( )

7.5 Zero Cost Operation

Zero cost operation is an optimization that may be done where source and destination modules are single and in source code (one of the modules must be in source code otherwise the PDU Router must create glue-code for the function call). For example an ECU with a COM module and a single CAN bus, the PduR_ComTransmit may directly call the CanIf_Transmit without any logic inside the PduR Module. The PDU Router becomes a macro layer.

This optimization is only possible where routing paths are of configuration class Pre-Compile.

[PDUR287] ⌈If PduRZeroCostOperation is set to true and all routing paths are of configuration class Pre-Compile; modules directly above or below the PDU Router may directly call each other without using PduR module functions. ⌋( )

[PDUR619] ⌈If PduRZeroCostOperation is set to true and at least one routing path is not of configuration class Pre-Compile; the PDU Router module configuration generator shall report an error. ⌋( )

7.6 Minimum Routing

Minimum routing is routing paths that are of configuration class Link-Time or Pre-Compile. The idea is to have a minimum set of routing paths that can be used for communication in case the Post-Build routing paths are erroneous. Usually TP communication is using minimum routing to enable diagnostics communication.

Use-case: Access to DCM to bring the ECU into programming mode even when the PDU Router routing tables are corrupted.

[PDUR285] ⌈The minimum routing table shall be separated from the PDU Router routing tables and shall only be configurable at pre-compile time or link-time. ⌋( )

[PDUR286] ⌈The PDU Router module shall always precede minimum routing over routing according to the normal configurable PDU Router routing tables. ⌋( )

7.7 Reentrancy of API calls

The reentrancy of API calls is generally specified for each API call. There is some extra general requirements on the reentrancy of the API calls:

[PDUR630] ⌈In case the API is reentrant and dependent on I-PDU ID the PDU Router shall not accept a new call to the same API while a former call is still ongoing. ⌋( )

[PDUR622] ⌈If an I-PDU ID that is in process and a module request to process an I-

PDU with the same I-PDU ID the PDU Router module shall serve the new request and PDUR_E_DUPLICATE_IPDU_ID shall be reported to DET, when PduRDevErrorDetect is enabled. ⌋( )

Example: The Com modules calls PduR_ComTransmit to be transported on the CanIf module, but the CanIf module has not confirmed the last request with the same I-PDU ID, the PduR will call CanIf_Transmit without waiting for a confirmation for the previous transmit request.

7.8 State Management

The state machine of the PDU Router module is depicted in Figure 5.

[PDUR644] ⌈Only one instance of the state machine shall exist in the PDUR module. ⌋( )

[PDUR324] ⌈The PDU Router module shall consist of three states, `PDUR_UNINIT`, `PDUR_REDUCED` and `PDUR_ONLINE`. ⌋( )

[PDUR325] ⌈The PDU Router module shall be in the state `PDUR_UNINIT` after power up the PDU Router module (i.e. before calling the PduR_Init function). ⌋( )

[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, e.g. Post-Build configuration not available. ⌋( )

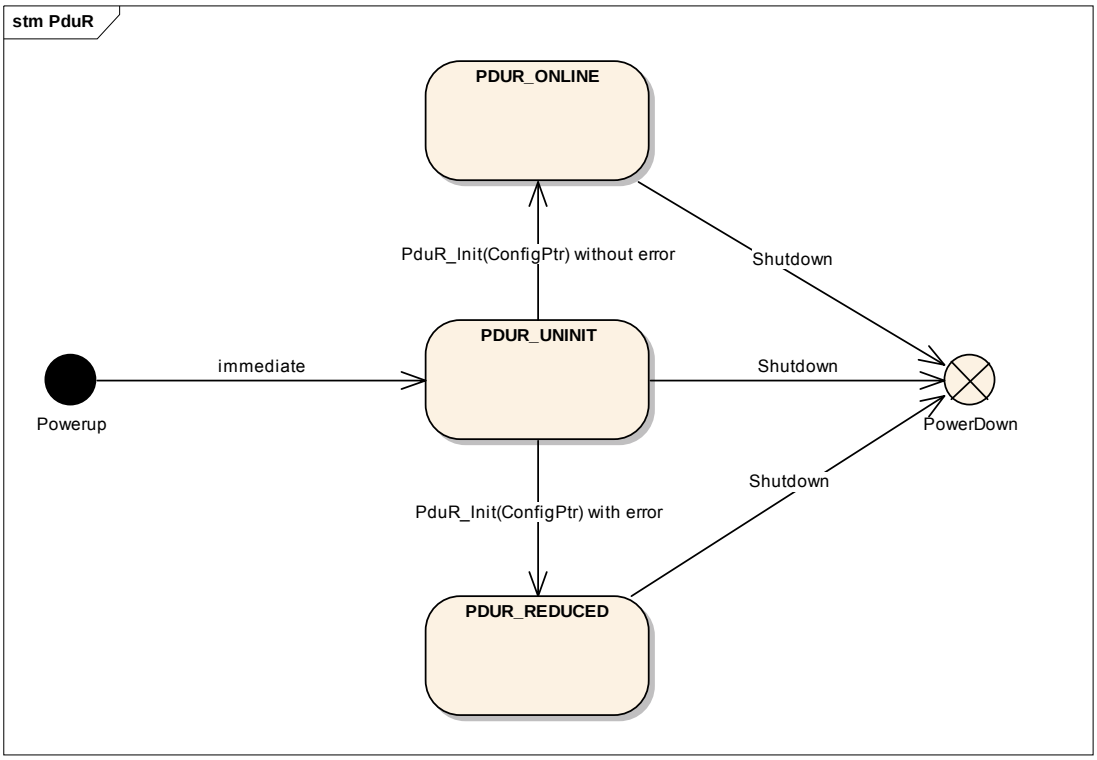

Figure 5: PDU Router states

[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 uninitialized state (`PDUR_UNINIT`). ⌋( )

[PDUR645] ⌈The PDU Router module shall release all buffers in the PduR_Init function. ⌋( )

[PDUR308] ⌈The function PduR_Init shall initialize all configured default value to the PDU transmit buffers. ⌋( )

7.9 Routing path groups

A routing path group is a group of I-PDUs that can be disabled and enabled during runtime. The group contains the destination I-PDUs and not the routing path itself. The reason is that it is desirable to enable disable I-PDUs for a specific bus. And a routing path can multicast an I-PDU to several busses.

Enabling and disabling of routing path groups is typically used by the BswM module.

[PDUR0714] ⌈If the I-PDU ID is used in an API that leads to a multicast (e.g. PduR_<Up>Transmit) and the I-PDU ID is disabled in all destination modules E_NOT_OK (if possible) shall be returned with no further action. ⌋( )

[PDUR0717] ⌈If the I-PDU ID used in an API is disabled in all destination modules E_NOT_OK (if possible) shall be returned with no further action. ⌋( )

[PDUR0715] ⌈Enabling of I-PDU routing path groups shall be immediate⌋( )

Example: A subsequent call to PduR_<Up>Transmit shall serve this I-PDU directly.

[PDUR646] ⌈The PDU Router shall immediately disable the routing path table, even though transmissions/receptions are ongoing. ⌋( )

Example: If CanTp is currently transmitting a multiple N-PDUs then PduR will reject further requests using this disabled I-PDU.

[PDUR663] ⌈All buffers used in a routing path group shall be cleared when a routing path group is disabled. ⌋( )

Example: If a gateway operation is made and the PDU Router module has buffered an I-PDU and is waiting for the destination communication module to call trigger transmit, the buffer is cleared and the buffer not available is returned to the destination communication interface.

7.10 Complex Device Driver Interaction

Besides the AUTOSAR Com and Dcm modules, Complex Device Drivers (CDD) are

also possible as upper layer modules for the PduR.

The PduR provides the unique transmit function PduR_<Cdd>Transmit for each upper layer CDD. When a callout function of the PduR is invoked from a lower layer module for a Pdu that is transmitted or received by a CDD, the PduR invokes the corresponding target function of the CDD.

To determine if a Pdu is transmitted or received by a CDD, the PduR has to examine the origin of the references to the Pdu list in the EcuC module. If the source Pdu of a routing path references a Pdu in the Pdu list that is also referenced by a CDD, the Pdu is transmitted by the CDD. If the destination Pdu of a routing path references a Pdu in the Pdu list that is also referenced by a CDD, the Pdu is received by the CDD.

A CDD can either require a communication interface API or it can require a transport protocol API but not both. The API functions provided by the PduR for the CDD interaction contain the CDD's name.

[PDUR504] ⌈ The PduR shall use the apiServicePrefix attribute of the CDD's vendor specific module definition (EcucModuleDef element) to replace the <Lo>, <Up>, and <LoTp> tags of the GenericComServices APIs (see Section 12). The CDD's vendor specific module definition can be indirectly accessed via the configuration parameter PduRBswModuleRef which references the top-level element of the concrete configuration of the CDD (i.e., EcucModuleConfigurationValues element) which references the CDD's vendor specific module definition (EcucModuleDef element). ⌋ ( )

To determine if a CDD requires a communication interface API or a transport protocol API, the PduR has to examine the references to the PDU list in the EcuC module. If all PDUs transmitted or received by a CDD are referenced by communication interface modules, the CDD requires a communication interface API. If all PDUs transmitted or received by a CDD are referenced by transport protocol modules, the CDD requires a transport protocol API.

7.11 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). These types of errors are reported to the DET module [5].

(b) errors and exceptions that are expected to occur also in production code. These types of errors are reported to the DEM module [6].

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

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Invalid configuration pointer	Development	PDUR_E_CONFIG_PTR_INVALID	0x00
API service used without module initialization or PduR_Init called in any state other than PDUR_UNINIT	Development	PDUR_E_INVALID_REQUEST	0x01
Invalid PDU identifier	Development	PDUR_E_PDU_ID_INVALID	0x02
TP module rejects a transmit request for a valid PDU identifier	Development	PDUR_E_TP_TX_REQ_REJECTED	0x03
If any of the parameter passed is out of respective ranges	Development	PDUR_E_PARAM_INVALID	0X04
A I-PDU ID is received that is already in process	Development	PDUR_E_DUPLICATE_IPDU_ID	0x06
I-PDU buffer is longer than expected	Development	PDUR_E_IPDU_TOO_LONG	0x07
If the routing table is invalid that is given to the PduR_EnableRouting or PduR_DisableRouting functions	Development	PDUR_E_ROUTING_PATH_GROUP_ID_INVALID	0x08
Loss of a PDU instance (FIFO flushed because of an overrun)	Development	PDUR_E_PDU_INSTANCES_LOST	0x0a
PDU Router initialization failed (PDU Router changed to PDUR_REDUCED state)	Development	PDUR_E_INIT_FAILED	0x0b
Pointer parameter is null. Note that specific API calls may disable this error	Development	PDUR_E_NULL_POINTER	0x09

」( )

[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. ⌋( )

7.12 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 7.11 and chapter 7.14. ⌋( )

[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 and PduR_GetVersionInfo shall report the error `PDUR_E_INVALID_REQUEST` via the DET when called, when PduRDevErrorDetect is enabled. ⌋(BSW00406)

7.13 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 (PduRDevErrorDetect) 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:

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

`ModuleId`	Module ID of the PDU Router: 51 decimal (see PDUR217)
`InstanceId`	0 (single instance module)
`ApiId`	ID of API which reports an error: Service ID defined in section 8.3
`ErrorId`	ID of detected development error: value according to section 7.11

[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 [6].⌋( )

[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].⌋( )

7.14 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, when PduRDevErrorDetect is enabled.⌋(BSW06103)

7.15 Debugging

[PDUR487] ⌈Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.⌋( )

[PDUR488] ⌈All type definitions of variables which shall be debugged, shall be accessible by the header file PduR.h.⌋( )

[PDUR489] ⌈The declaration of variables in the header file shall be such that it is possible to calculate the size of the variables by C-"sizeof".⌋( )

[PDUR490] ⌈Variables available for debugging shall be described in the respective Basic Software Module Description⌋ ( )

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] ⌈

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
	NotifResultType
	PduIdType
	PduInfoType
	PduLengthType
	RetryInfoType
	TPParameterType
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ ( )

8.2 Type definitions

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

8.2.1 PduR_PBConfigType

The post-build-time configuration fulfills two functionalities:

- Post-build selectable, where more than one configuration is located in the

ECU, and one is selected at init of the PDU Router module

- Post-build loadable, where one configuration is located in the ECU. This configuration may be reprogrammed after compile-time

Basically there is no restriction to mix both selectable and loadable. Typically the post-build loadable is located in its own flash sector where it can be reprogrammed without affecting other modules/applications.

[PDUR0743] ⌈

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

⌋ (BSW00400, BSW00438)

[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). ⌋ ( )

8.2.2 PduR_ConfigIdType

This type is returned by the PduR_GetConfigurationId API.

[PDUR0771] ⌈

Name:	`PduR_PBConfigIdType`
Type:	`--`
Description:	Identification of the post-build configuration currently used for routing I-PDUs. An ECU may contain several configurations (post-build selectable), each have unique Id.

⌋ ( )

8.2.3 PduR_RoutingPathGroupIdType

The routing path group ID is used for identifying a specific group of routing path destinations. The reason is that the destinations of a 1:n routing path typically belong to more than one bus, and it shall be possible to enable/disable routing per bus.

Therefore a routing path group separates 1:n routing paths into 1:1 paths.

[PDUR654] ⌈

Name:	`PduR_RoutingPathGroupIdType`
Type:	`uint16`
Description:	Identification of a Routing Table

⌋ ( )

8.2.4 PduR_StateType

[PDUR0742] ⌈

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

⌋ (BSW00441)

8.3 Function definitions

8.3.1 General functions provided by the PDU Router

8.3.1.1 PduR_Init

[PDUR334] ⌈

Service name:	PduR_Init

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

Service ID[hex]:	0xf0
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	ConfigPtr	Pointer to post build configuration
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the PDU Router	

」(BSW101, BSW00358, BSW00414)

[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` to the DET if the initialization of the PDU Router module failed.」( )

Integration note: To avoid problems calling the PDU Router module uninitialized it is important that the PDU Router module is initialized before interfaced modules.

[PDUR0709]「After initialization all I-PDU routing groups shall be enabled according enable at start configuration parameter.」( )

[PDUR0776]「If the ConfigPtr parameter is null the PDUR_E_NULL_POINTER shall not be sent to the DET module.」( )

The rationale of above requirement is that if the PduR module is implemented as pre-compile variant the pointer parameter will be null.

8.3.1.2 PduR_GetVersionInfo

[PDUR338]「

Service name:	PduR_GetVersionInfo

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

Service ID[hex]:	0xf1
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters	None

(inout):		
Parameters (out):	versionInfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

⌋(BSW00407, BSW00411)

[PDUR234] ⌈The function PduR_GetVersionInfo shall return the version information of this module in the passed argument *versionInfo. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers.⌋( )

8.3.1.3 PduR_GetConfigurationId

[PDUR341] ⌈

Service name:	PduR_GetConfigurationId	
Syntax:	`PduR_PBConfigIdType PduR_GetConfigurationId( void )`	
Service ID[hex]:	0xf2	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	PduR_PBConfigIdType	Identifier of the post-build time configuration
Description:	Returns the unique identifier of the post-build time configuration of the PDU Router	

⌋(BSW06097)

[PDUR280] ⌈The function PduR_GetConfigurationId shall return the unique identifier of the post-build time configuration of the PDU Router module.⌋( )

8.3.1.4 PduR_EnableRouting

[PDUR615] ⌈

Service name:	PduR_EnableRouting

Syntax:	`void PduR_EnableRouting(`
	`  PduR_RoutingPathGroupIdType id`
	`)`

Service ID[hex]:	0xf3
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	id Identification of the routing path group. Routing path groups are defined in the PDU router configuration.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Enables a routing path table.

」(BSW06120)

[PDUR647]「If the routing path group id does not exist, then the PDU Router module shall return with no action.」( )

[PDUR648]「If the routing path group id does not exist and the PduRDevErrorDetect is enabled, the PDU Router module shall report PDUR_E_ROUTING_PATH_GROUP_ID_INVALID.」( )

8.3.1.5 PduR_DisableRouting

[PDUR617]「

Service name:	PduR_DisableRouting

Syntax:	`void PduR_DisableRouting(`
	`  PduR_RoutingPathGroupIdType id`
	`)`

Service ID[hex]:	0xf4
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	id Identification of the routing path group. Routing path groups are defined in the PDU router configuration.
Parameters	None

(inout):	
Parameters (out):	None
Return value:	None
Description:	Disables a routing path table.

⌋(BSW06120)

[PDUR0716] ⌈If the routing path group id does not exist, then the PDU Router module shall return with no action.⌋( )

[PDUR649] ⌈If the routing path table id does not exist and the PduRDevErrorDetect is enabled, the PDU Router module shall report PDUR_E_ROUTING_PATH_GROUP_ID_INVALID.⌋( )

8.4 Scheduled functions

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

8.5 Expected Interfaces

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

8.5.1 Mandatory Interfaces

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

[PDUR423] ⌈

API function	***Description***
<Provider:Lo>_Transmit	Requests the sending of a PDU.
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.

⌋( )

Since the API have now a generic approach it is not relevant to list all possible

modules here. I

8.5.2 Optional Interfaces

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

[PDUR424] ⌈

API function	*Description*
<Provider:Lo>_CancelTransmit	Requests cancellation of a specific I-PDU in a lower layer communication interface module.
<Provider:LoTp>CancelReceive	Requests cancellation of an I-PDU reception in a lower layer transport protocol module.
<Provider:LoTp>CancelTransmit	Requests cancellation of a specific I-PDU in a lower layer transport protocol module.
<Provider:LoTp>_ChangeParameter	Request to change a specific transport protocol parameter (e.g. block-size).
<Provider:UpTp>_CopyRxData	This function is called when transport protocol module have data to copy to the receiving module. Several calls may be made during one transportation of an I-PDU.
<Provider:UpTp>_CopyTxData	This function is called by the transport protocol module to query the transmit data of an I-PDU segment. Each call to this function copies the next part of the transmit data until TpDataState indicates TP_DATARETRY. In this case the API restarts to copy the data beginning at the location indicated by TpDataCnt.
<Provider:UpTp>_StartOfReception	This function will be called by the transport protocol module at the start of receiving an I-PDU. The I-PDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdulength equal to 0.
<Provider:UpTp>_TpRxIndication	Called by the transport protocol module after an I-PDU has been received successfully or when an error occurred. It is also used to confirm cancellation of an I-PDU.
<Provider:UpTp>_TpTxConfirmation	This function is called by a transport protocol module after the I-

	PDU has been transmitted on its network, the result will reveal if the transmission was successful or not.
Det_ReportError	Service to report development errors.

⌋( )

8.5.3 Configurable interfaces definitions for interaction with upper layer module

Since the API description now has a generic approach, the service ids of the upper layer API functions are generic as well. The generic service id of an upper layer API function consists of an upper layer module id and a service base id.

[PDUR0780] ⌈For the generic service Ids of the interfaces defined for interaction with upper layer modules, the following upper layer module id <UpModId> shall be used:

Upper layer module	<UpModId>
Com	0x8
Dcm	0x9
IpduM	0xa
Dbg	0xb
CDD for interface interaction	0xc
CDD for transport protocol interaction	0xd

⌋( )

Example: The service id of PduR_ComTransmit is 0x89.

8.5.3.1 PduR_<Up>Transmit

[PDUR406] ⌈

Service name:	PduR_<User:Up>Transmit

Syntax:	`Std_ReturnType PduR_<User:Up>Transmit(`
	`  PduIdType id,`
	`  PduInfoType* info`
	`)`

Service ID[hex]:	0x<UpModId>9
Sync/Async:	Asynchronous
Reentrancy:	Reentrant

Parameters (in):	id	Identification of the I-PDU.
	info	Length and pointer to the buffer of the I-PDU
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK - request is accepted by the destination module. E_NOT_OK - request is not accepted by the destination module.
Description:	Requests transmission of an I-PDU.	

⌋ ( )

8.5.3.2 PduR_<Up>CancelTransmit

[PDUR0769] ⌈

Service name:	PduR_<User:Up>CancelTransmit	

Syntax:	`Std_ReturnType PduR_<User:Up>CancelTransmit(`
	`  PduIdType id`
	`)`	

Service ID[hex]:	0x<UpModId>a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	id	Identification of the I-PDU to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request is accepted by the destination module E_NOT_OK: request is not accepted by the destination module.
Description:	Request for cancellation of an ongoing transmission of an I-Pdu in transport protocol or communication interface.	

⌋ ( )

8.5.3.3 PduR_<Up>ChangeParameter

[PDUR482] ⌈

Service name:	PduR_<User:Up>ChangeParameter
Syntax:	`Std_ReturnType PduR_<User:Up>ChangeParameter(`

	PduIdType id, TPParameterType parameter, uint16 value)
Service ID[hex]:	0x<UpModId>b
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	id: Identification of the I-PDU to which the parameter the request shall affect. parameter: The selected parameter that the request shall changed. value: The value that the request shall change to.
Parameters (inout):	None
Parameters (out):	None
Return value:	Std_ReturnType: E_OK: request is accepted. E_NOT_OK: request is not accepted.
Description:	Request to change a specific transport protocol parameter (e.g. block-size). The affected transport protocol module is selected using the I-PDU ID.

⌋ ( )

8.5.3.4 PduR_<Up>CancelReceive

[PDUR0767] ⌈

Service name:	PduR_<User:Up>CancelReceive
Syntax:	Std_ReturnType PduR_<User:Up>CancelReceive(PduIdType id)
Service ID[hex]:	0x<UpModId>c
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	id: Identification of the I-PDU.
Parameters (inout):	None
Parameters (out):	None
Return value:	Std_ReturnType: E_OK - request accepted (but not yet performed). E_NOT_OK - request not accepted (e.g. cancellation not

		possible).
Description:	Requests for cancellation of an ongoing reception of an I-Pdu in transport protocol.	

⌋ ( )

8.5.4 Configurable interfaces definitions for lower layer communication interface module interaction

Since the API description now has a generic approach, the service ids of the lower layer communication interface API functions are generic as well. The generic service id of lower layer communication interface API function consists of lower layer interface module id and a service base id.

[PDUR0781] ⌈For the generic service Ids of the interfaces defined for interaction with lower layer communication interface modules, the following lower layer interface module id <LoIfModId> shall be used:

Lower layer communication interface module	<LoIfModId>
CanIf/TTCanIf	0x0
CanNm	0x1
IpduM	0x2
FrIf	0x3
FrNm	0x4
LinIf	0x5
SoAdIf	0x6
CDD If	0x7

⌋ ( )

Example: The service id of PduR_FrIfRxIndication is 0x31.

8.5.4.1 PduR_<Lo>RxIndication

[PDUR362] ⌈

Service name:	PduR_<Lo>RxIndication

Syntax:	`void PduR_<Lo>RxIndication(`
	`  PduIdType RxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`

Service ID[hex]:	0x<LoModId>1	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	ID of the received I-PDU.
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication module.	

⌋ ( )

8.5.4.2 PduR_<Lo>TxConfirmation

[PDUR365] ⌈

Service name:	PduR_<Lo>TxConfirmation	
Syntax:	`void PduR_<Lo>TxConfirmation( PduIdType TxPduId )`	
Service ID[hex]:	0x<LoModId>2	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the I-PDU that has been transmitted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication module confirms the transmission of an I-PDU.	

⌋ ( )

8.5.4.3 PduR_<Lo>TriggerTransmit

[PDUR369] ⌈

Service name:	PduR_<Lo>TriggerTransmit
Syntax:	`Std_ReturnType PduR_<Lo>TriggerTransmit(`

	PduIdType TxPduId, PduInfoType* PduInfoPtr)	
Service ID[hex]:	0x<LoModId>3	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

⌋( )

8.5.5 Configurable interfaces definitions for lower layer transport protocol module interaction

Since the API description now has a generic approach, the service ids of the lower layer transport protocol API functions are generic as well. The generic service id of lower layer transport protocol API function consists of lower layer transport protocol module id and a service base id.

[PDUR0782] ⌈For the generic service Ids of the interfaces defined for interaction with lower layer transport protocol modules, the following lower layer transport protocol module id <LoTpModId> shall be used:

Lower layer transport protocol module	<LoTpModId>
CanTp	0x0
SAE J1939	0x1
FrTp	0x3
LinTp	0x5

SoAdTp	0x6
CDD Tp	0x7

⌋( )

Example: The service id of PduR_FrTpRxIndication is 0x35.

8.5.5.1 PduR_<LoTp>CopyRxData

[PDUR512] ⌈

Service name:	PduR_<User:LoTp>CopyRxData	

Syntax:	`BufReq_ReturnType PduR_<User:LoTp>CopyRxData(`
	`  PduIdType id,`
	`  PduInfoType* info,`
	`  PduLengthType* bufferSizePtr`
	)	

Service ID[hex]:	0x<LoTpModId>4	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the received I-PDU.
	info	Pointer to the buffer (SduDataPtr) and its length (SduLength) containing the data to be copied by PDU Router module in case of gateway or upper layer module in case of reception.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer after data has been copied.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successfully. BUFREQ_E_NOT_OK: Buffer request not successful. Buffer cannot be accessed. BUFREQ_E_BUSY: Temporarily no buffer available. It's up the requestor to retry request for a certain time.
Description:	This function is called when a transport protocol module has data to copy for the receiving module. Several calls may be made during one transportation of an I-PDU. The service shall provide the currently available buffer size when invoked with info.SduLength equal to 0.	

⌋( )

8.5.5.2 PduR_<LoTp>RxIndication

[PDUR375] ⌈

Service name:	PduR_<User:LoTp>RxIndication	

Syntax:	`void PduR_<User:LoTp>RxIndication(`
	`  PduIdType id,`
	`  NotifResultType result`
	`)`	

Service ID[hex]:	0x<LoTpModId>5	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the received I-PDU.
	result	Result of the reception.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Called by the transport protocol module after an I-PDU has been received successfully or when an error occurred. It is also used to confirm cancellation of an I-PDU.	

⌋ ( )

8.5.5.3 PduR_<LoTp>StartOfReception

[PDUR507] ⌈

Service name:	PduR_<User:LoTp>StartOfReception	

Syntax:	`BufReq_ReturnType PduR_<User:LoTp>StartOfReception(`
	`  PduIdType id,`
	`  PduLengthType TpSduLength,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Service ID[hex]:	0x<LoTpModId>6	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the I-PDU.
	TpSduLength	Total length of the PDU to be received.

Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
Return value:	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted. RxBufferSizePtr indicates the available receive buffer. BUFREQ_E_BUSY: Currently no buffer of the requested size is available. RxBufferSizePtr remains unchanged. Connection has been rejected. BUFREQ_E_NOT_OK: Connection has been rejected. RxBufferSizePtr remains unchanged. BUFREQ_E_OVFL: No Buffer of the required length can be provided.
Description:	This function will be called by the transport protocol module at the start of receiving an I-PDU. The I-PDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSduLength equal to 0.	

⌋ ( )

8.5.5.4 PduR_<LoTp>CopyTxData

[PDUR518] ⌈

Service name:	PduR_<User:LoTp>CopyTxData	

Syntax:	`BufReq_ReturnType PduR_<User:LoTp>CopyTxData(`
	`  PduIdType id,`
	`  PduInfoType* info,`
	`  RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	)	

Service ID[hex]:	0x<LoTpModId>7	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the transmitted I-PDU.
	info	Provides destination buffer and the number of bytes to copy. In case of gateway the PDU Router module will copy,

		otherwise the source upper layer module will copy the data. If not enough transmit data is available, no data is copied. The transport protocol module will retry. A copy size of 0 can be used to indicate state changes in the retry parameter.
	retry	This parameter is used to retransmit data because problems during the last service call. If the I-PDU is transmitted from a local module (e.g. DCM) the PDU router module will just forward the parameter value without check. If the I-PDU is gatewayed from another bus, the PDU Router module will make the following interpretation: If the transmitted TP I-PDU does not support the retry feature a NULL_PTR is provided. It indicates that the copied transmit data can be removed from the buffer after it has been copied. If the retry feature is used by the Tx I-PDU, RetryInfoPtr must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that have been copied so far are confirmed and can be removed from the TP buffer. Data copied by this API call are excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy already copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset of the first byte to be copied by the API call.
Parameters (inout):	None	
Parameters (out):	availableDataPtr	Indicates the remaining number of bytes that are available in the PduR Tx buffer. AvailableTxDataCntPtr can be used by TP modules that support dynamic payload lengths (e.g. Iso FrTp) to determine the size of the following CFs.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer

		completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. TP layer might retry later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	This function is called by the transport protocol module to query the transmit data of an I-PDU segment. Each call to this function copies the next part of the transmit data until TpDataState indicates TP_DATARETRY. In this case the API restarts to copy the data beginning at the location indicated by TpTxDataCnt. The service shall provide the size of the remaining data when invoked with info.SduLength equal to 0.	

⌋ ( )

8.5.5.5 PduR_<LoTp>TxConfirmation

[PDUR381] ⌈

Service name:	PduR_<User:LoTp>TxConfirmation	

Syntax:	`void PduR_<User:LoTp>TxConfirmation(`
	`  PduIdType id,`
	`  NotifResultType result`
	`)`	

Service ID[hex]:	0x<LoTpModId>8	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the transmitted I-PDU.
	result	Result of the transmission of the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by a transport protocol module after the I-PDU has been transmitted on its network, the result will reveal if the transmission was successful or not.	

⌋ ( )

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.

Note: The sequence diagrams of the I-PDU Multiplexer are shown in [8]. Depending on the interaction scenario the IpduM has to be considered as an upper layer or a lower layer module of the PDU Router.

Note: The diagrams in this chapter are to show specific use-cases. They are not requirements for an implementation of the PDU Router module.

9.1 I-PDU Reception

The reception of an I-PDU received from a communication interface module or from transport protocol module and forwarded to the COM module.

Note that the PDU Router is not the only customer for the communication interface modules and I-PDUs. Other modules such as NM and TP modules receive PDUs directly from the communication interface modules.

9.1.1 CanIf module I-PDU reception

Following Figure 6 shows reception of I-PDU from the CanIf module to the COM module.

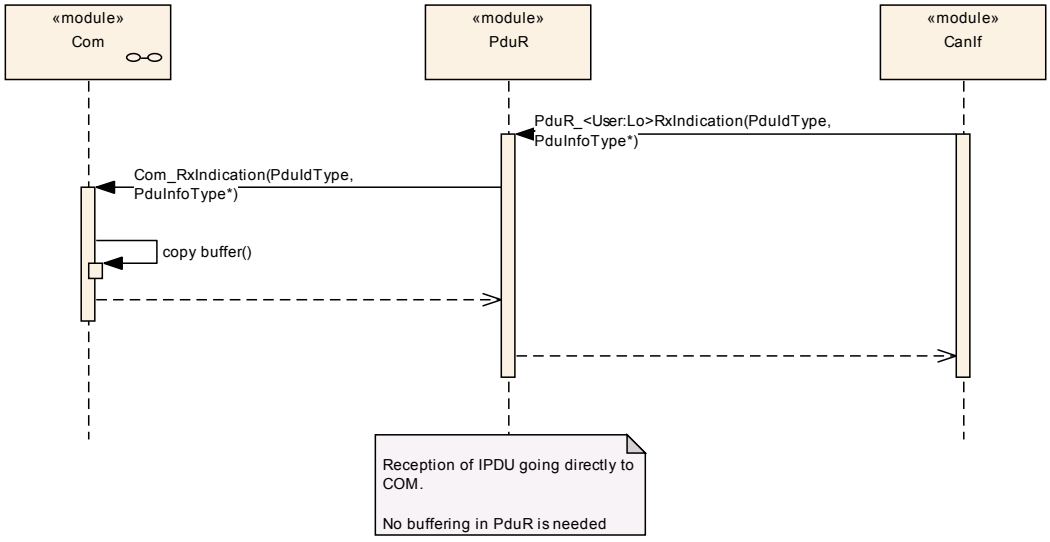

Figure 6: - CanIf I-PDU reception

9.1.2 FrIf module I-PDU reception

Following Figure 7 shows reception of I-PDU from the FrIf module to the COM module.

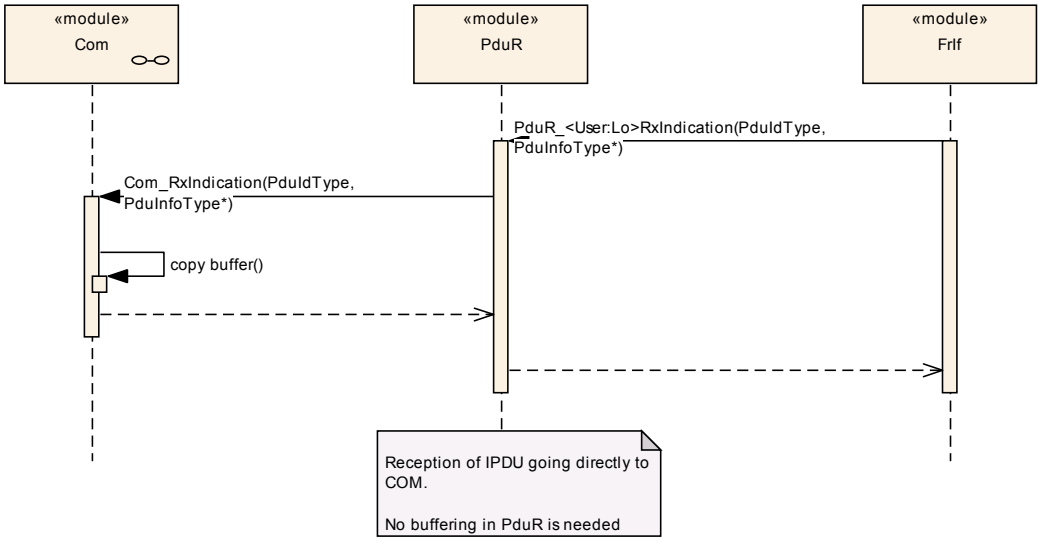

Figure 7: - FrIf I-PDU reception

9.1.3 LinIf module reception of I-PDU

Following Figure 8 shows reception of I-PDU from the LinIf module to the COM module.

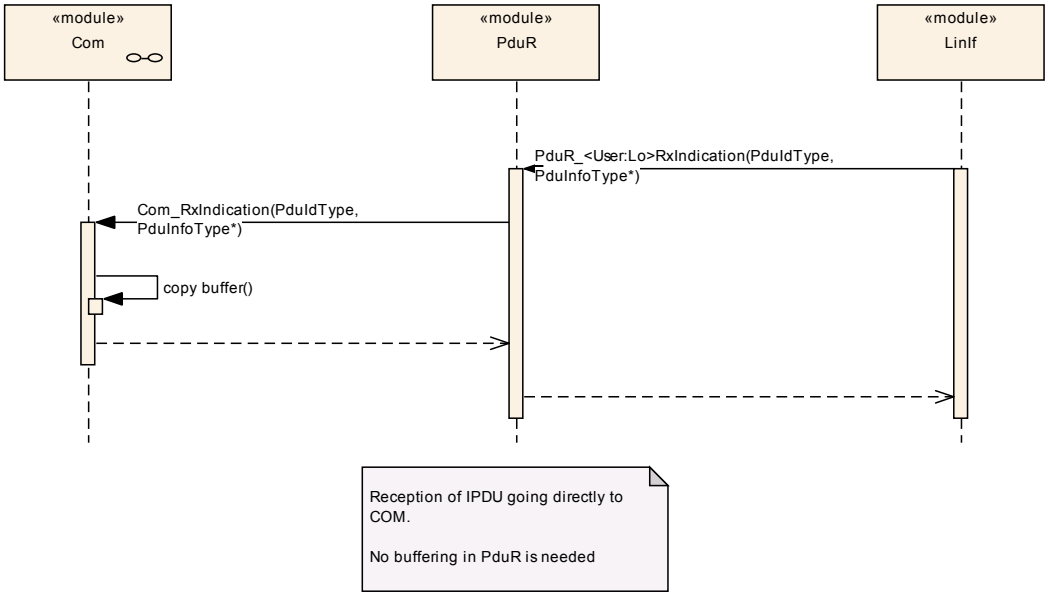

Figure 8: - LinIf I-PDU reception

9.1.4 CanTp module reception of I-PDU

Following Figure 9 shows reception of I-PDU from the CanTp module to the DCM module. The reception is made using the transport protocol APIs.

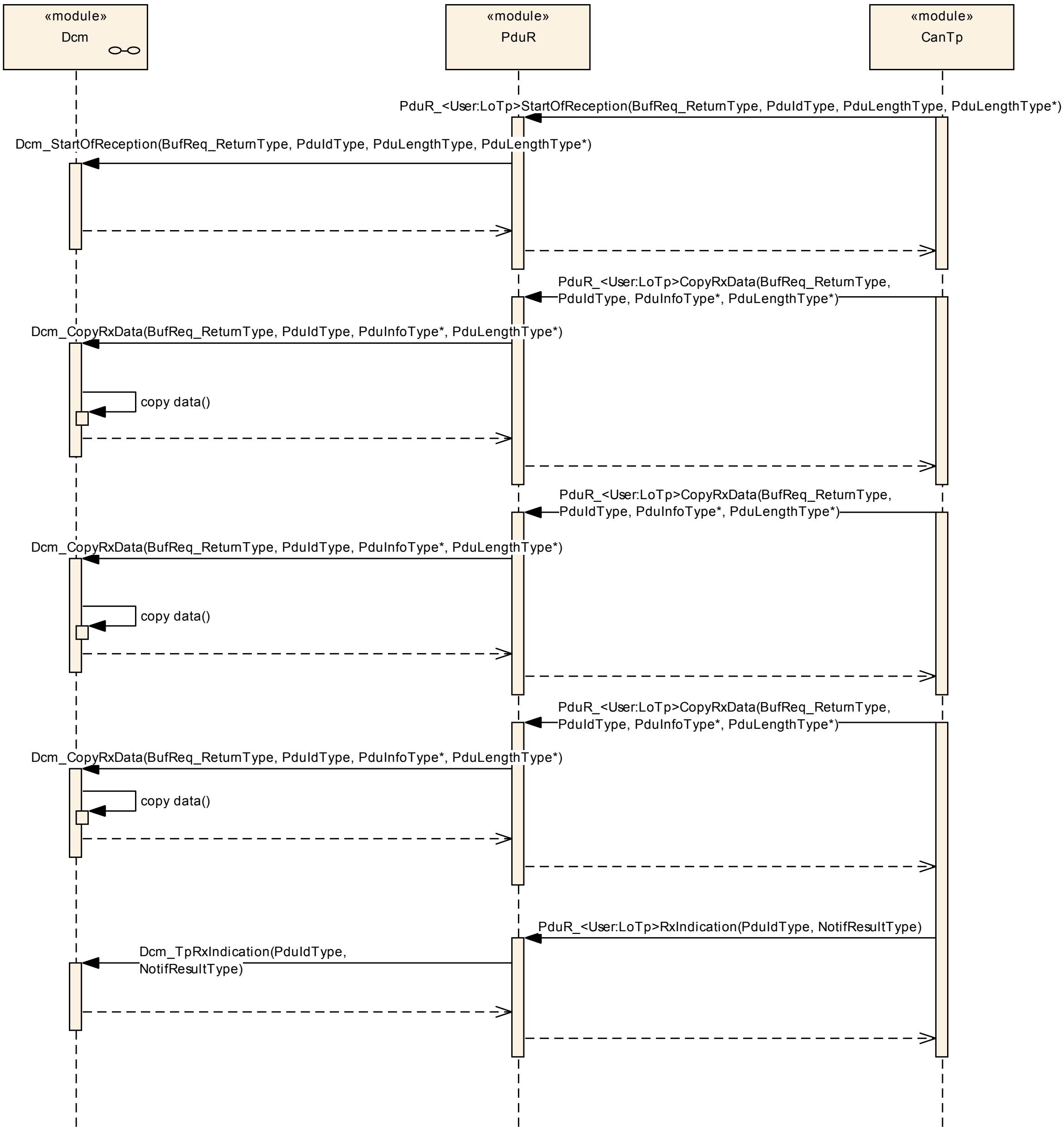

Figure 9 - CanTp I-PDU reception

9.2 I-PDU transmission

The transmission of an I-PDU transmitted from the COM module to a communication interface module or a transport protocol module.

9.2.1 CanIf module transmission of I-PDU

Following Figure 10 shows transmission of I-PDU from the COM module to the CanIf module.

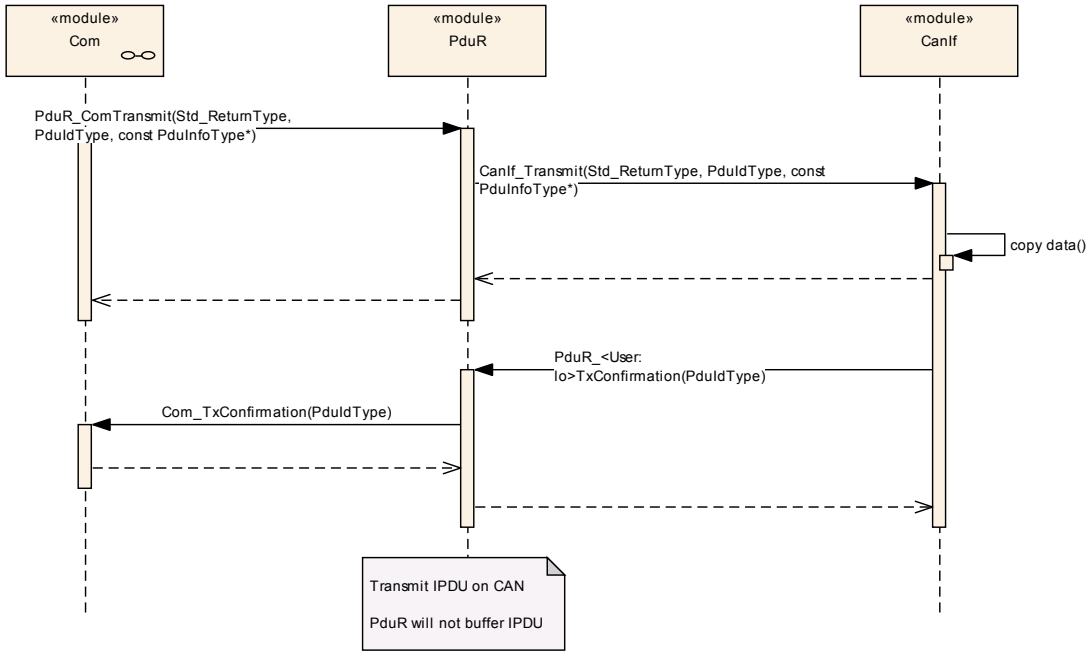

Figure 10: - CanIf I-PDU transmission

9.2.2 FrIf module transmission of I-PDU

Following Figure 11 shows transmission of I-PDU from the COM module to the FrIf module using trigger transmit.

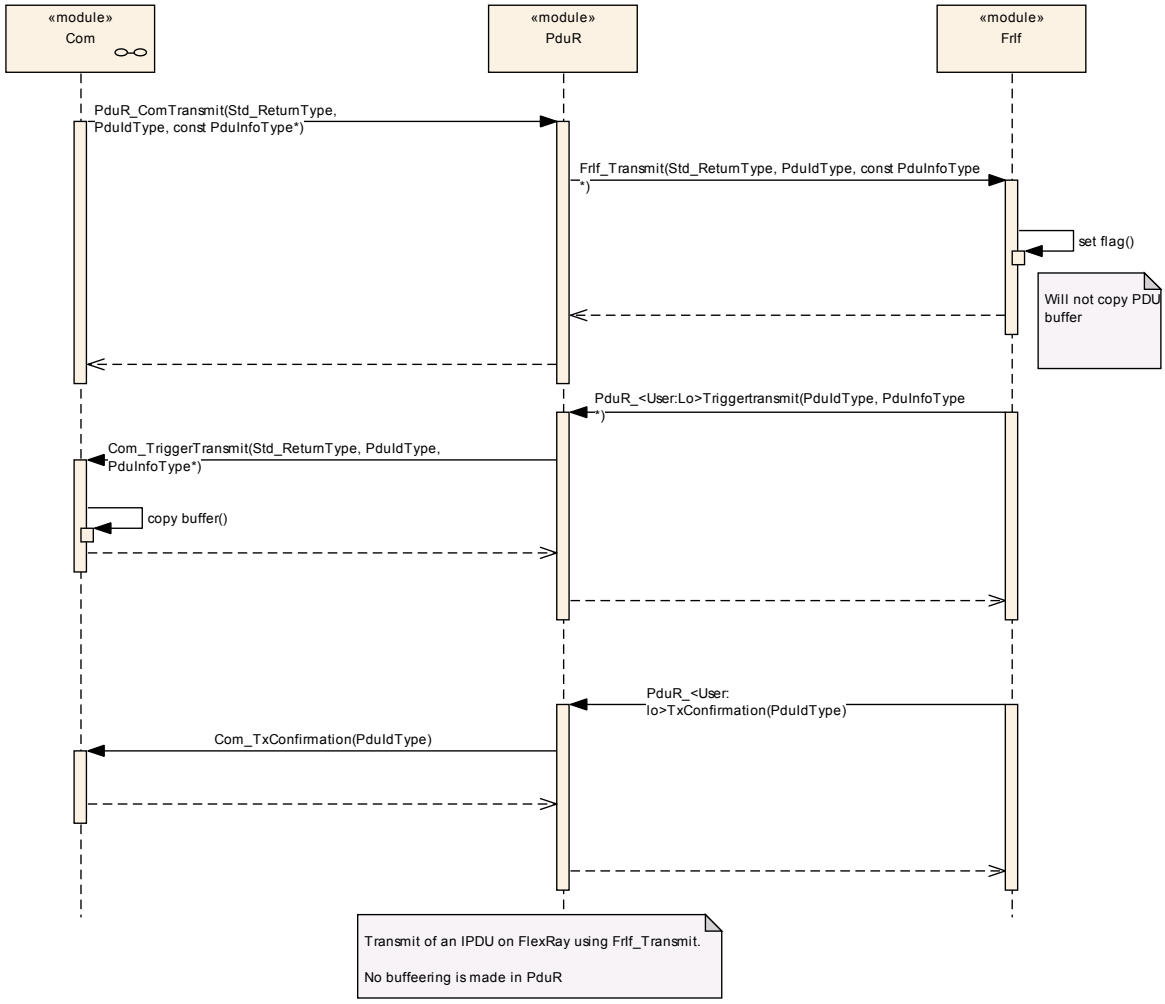

Figure 11: - FrIf I-PDU transmission

9.2.3 LinIf module transmission of I-PDU

Following Figure 12 shows transmission of I-PDU from the COM module to the LinIf module using transmit and later trigger transmit functions. In this case the I-PDU is a LIN sporadic frame.

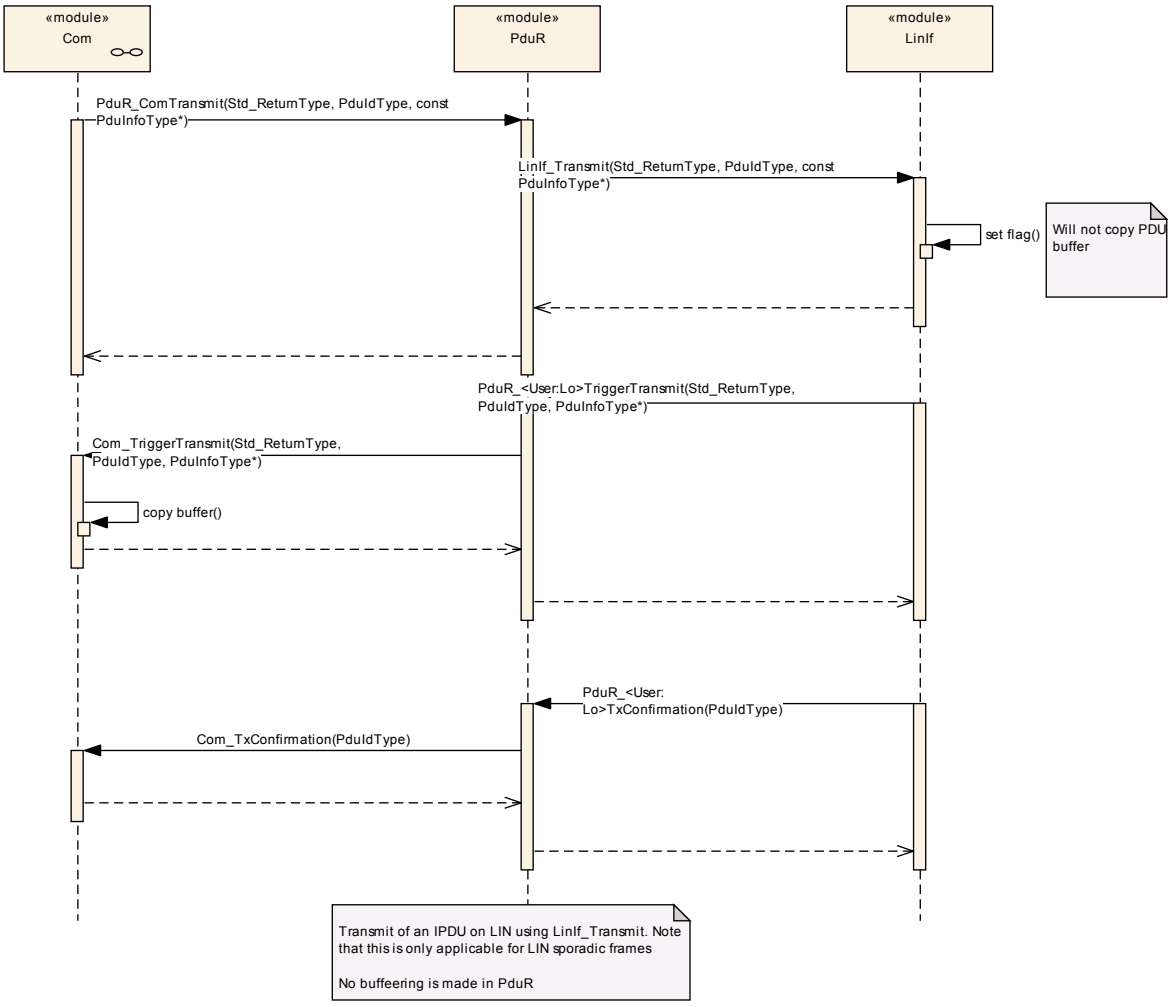

Figure 12: - LinIf I-PDU transmission (LIN sporadic frame)

Following Figure 13 shows transmission of I-PDU from the COM module to the LinIf module using trigger transmit. In this case the I-PDU is all other types except LIN sporadic frame.

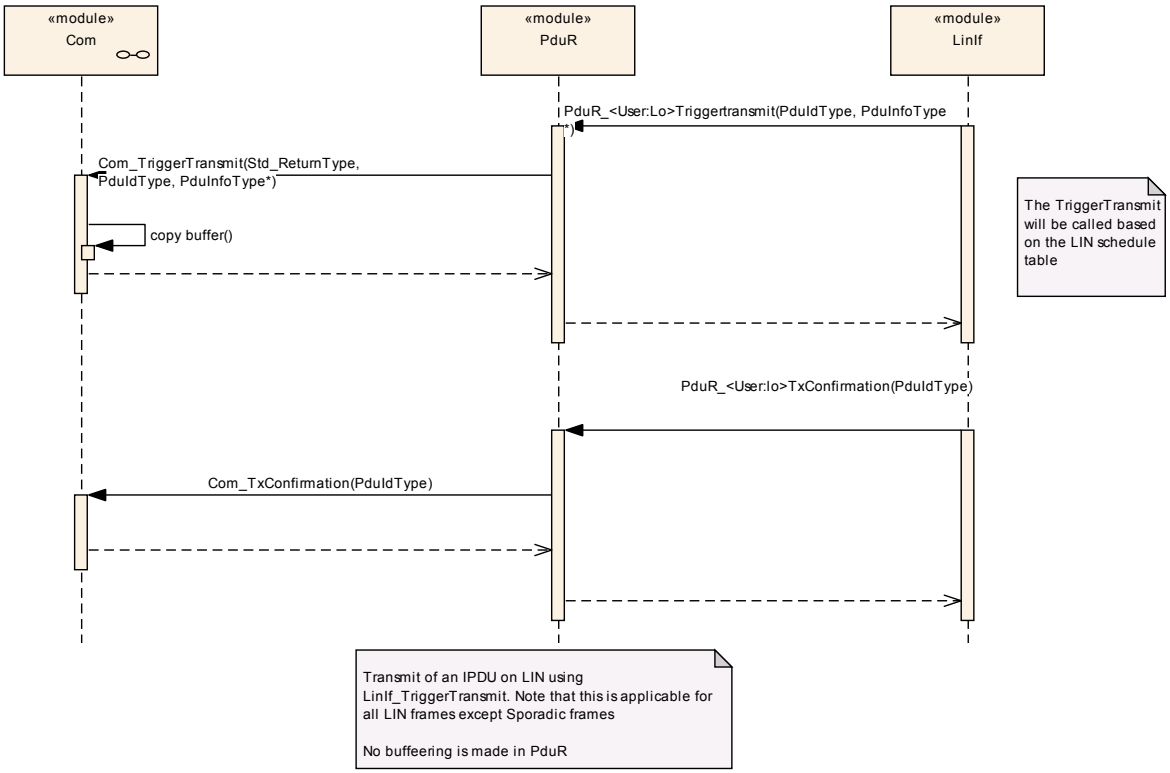

Figure 13: - LinIf I-PDU transmission (non LIN sporadic frame)

9.2.4 CanTp module transmission of I-PDU

Following Figure 14 shows transmission of I-PDU from the DCM module to the CanTp module using the transport protocol API.

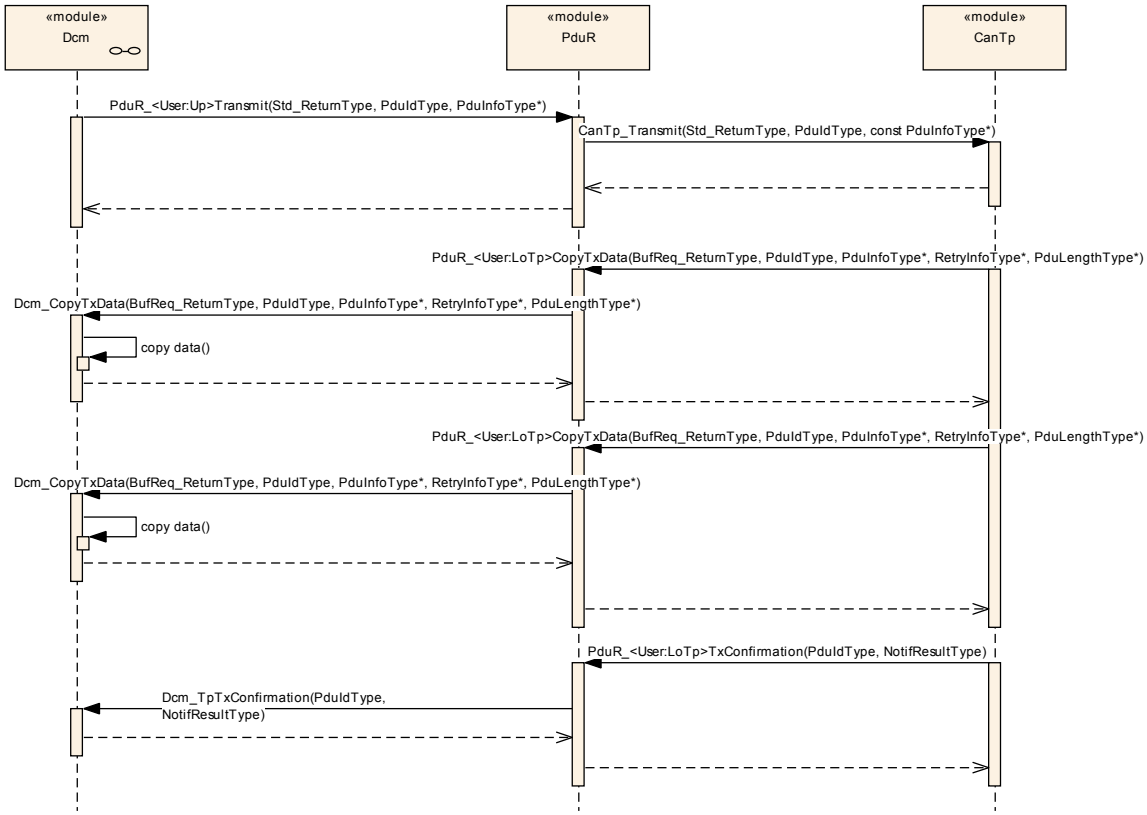

Figure 14: – CanTp I-PDU transmission

9.2.5 Multicast transmission of I-PDU on transport protocol modules

Following Figure 15 shows transmission of I-PDU from the DCM module to the CanTp, FrTp and LinTp (LinIf includes the transport protocol module) module using the transport protocol API.

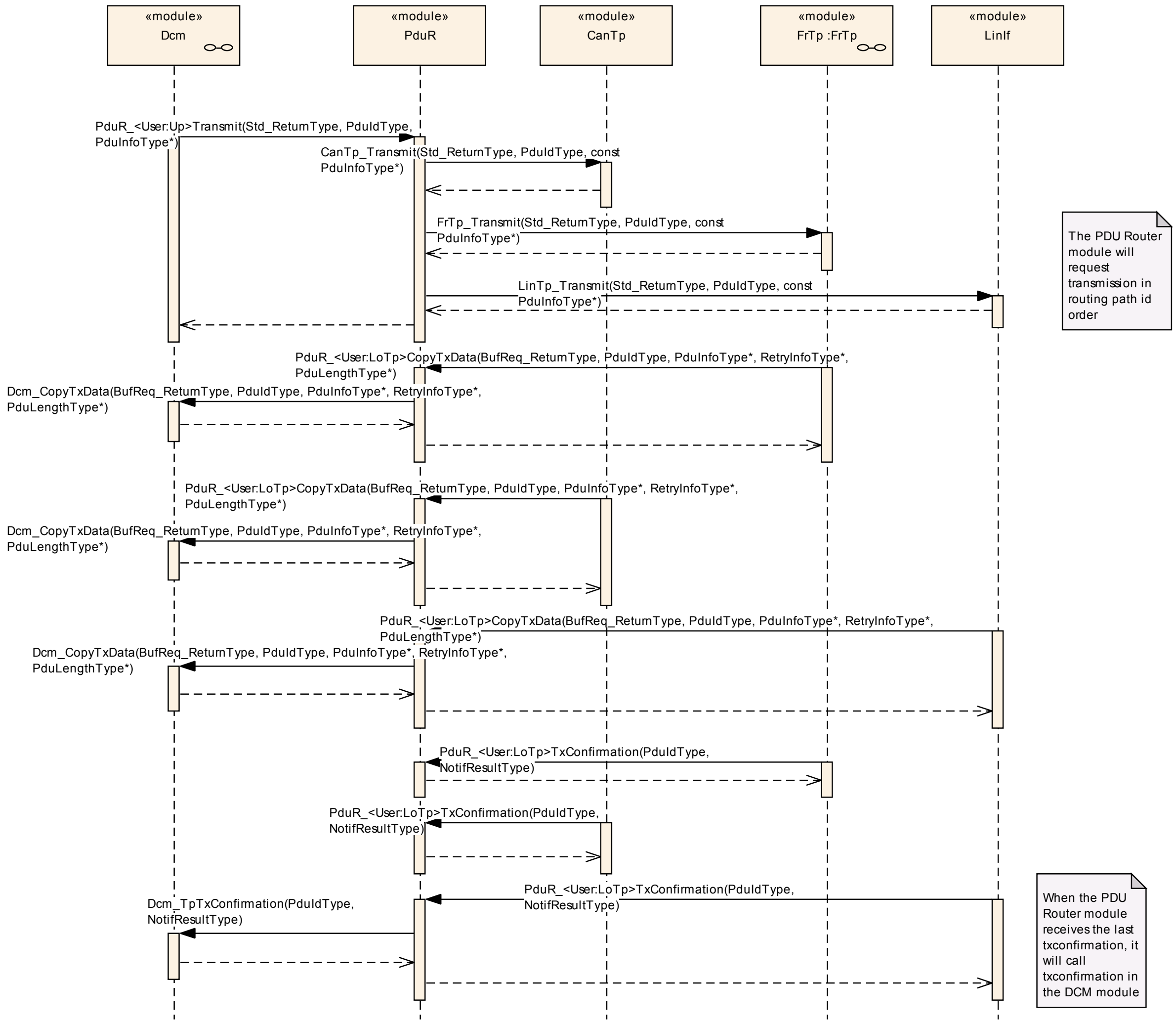

Figure 15 – I-PDU transmission on transport protocol on CAN, FlexRay and LIN

9.3 Gateway of I-PDU

Following use-cases shows how the PDU Router modules will gateway I-PDUs.

9.3.1 Gateway between two CanIfs

Following Figure 16 shows how an I-PDU is gatewayed between two CAN networks (CAN1 and CAN2) using CanIf.

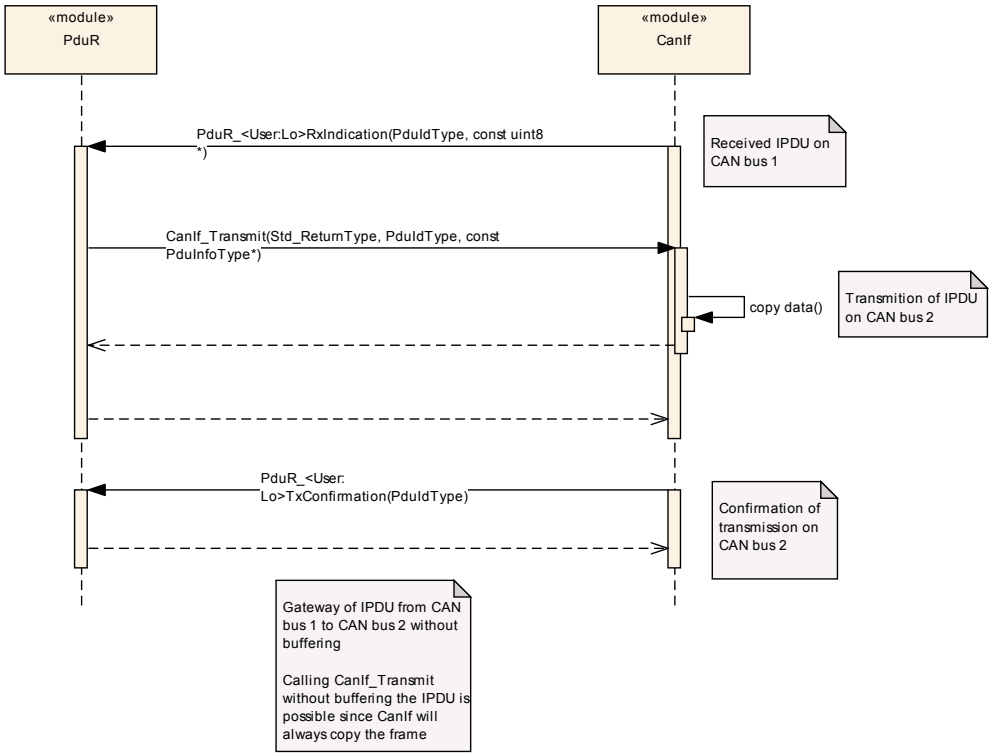

Figure 16: – Gateway of I-PDU from CAN1 to CAN2

9.3.2 Gateway from CAN to FlexRay

Following Figure 17 shows how an I-PDU is gatewayed between CAN and FlexRay, using trigger transmit (with buffering and without buffering).

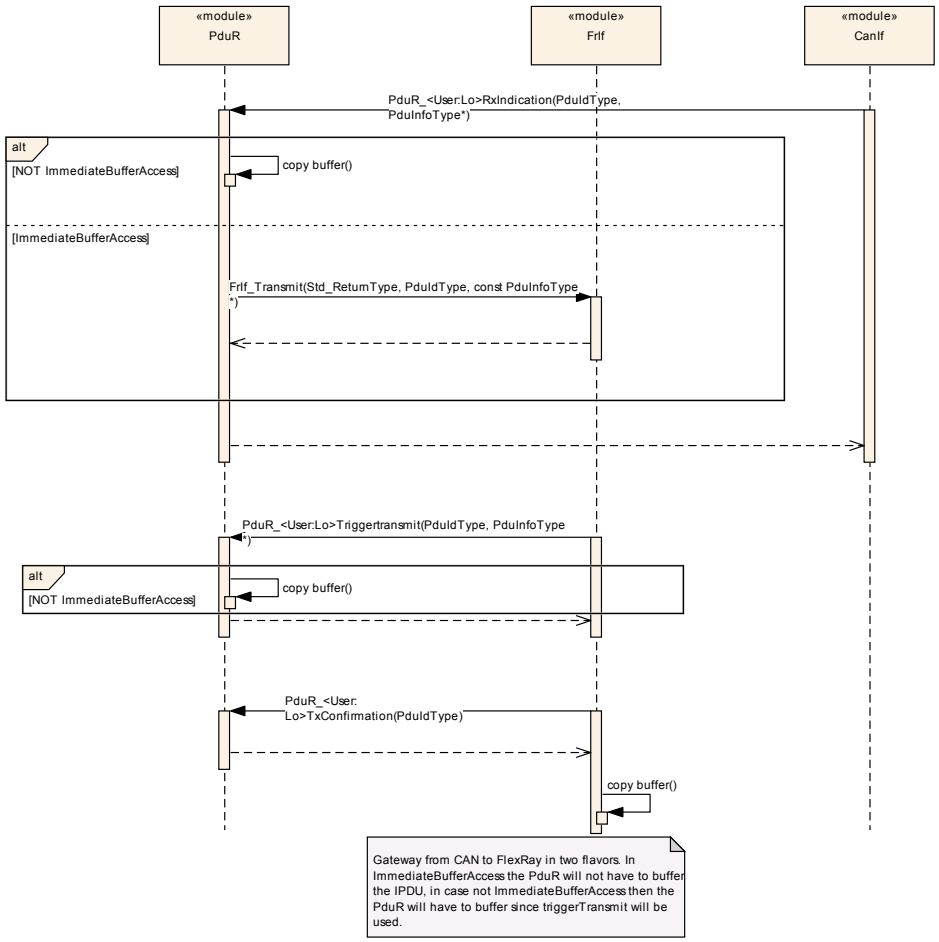

Figure 17: – Gateway of I-PDU from CAN to FlexRay

9.3.3 Gateway from CAN to LIN

Following Figure 18 shows how an I-PDU is gatewayed from CAN to LIN, using trigger transmit (LIN sporadic frame and all other LIN frame types).

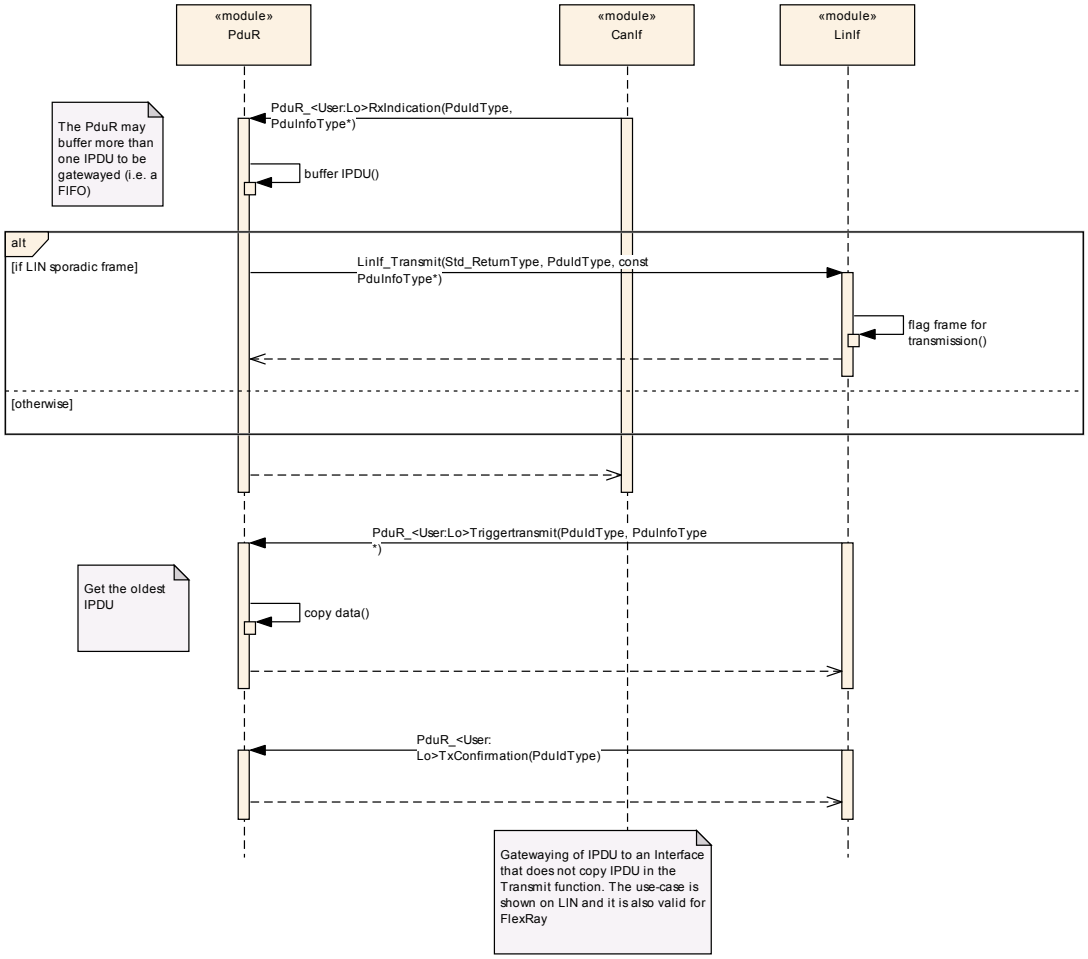

Figure 18: – Gateway of I-PDU from CAN to LIN

9.3.4 Gateway from CAN to CAN and received by the COM module

Following Figure 19 shows how an I-PDU is gatewayed from CAN1 to CAN2 and also received locally by the COM module.

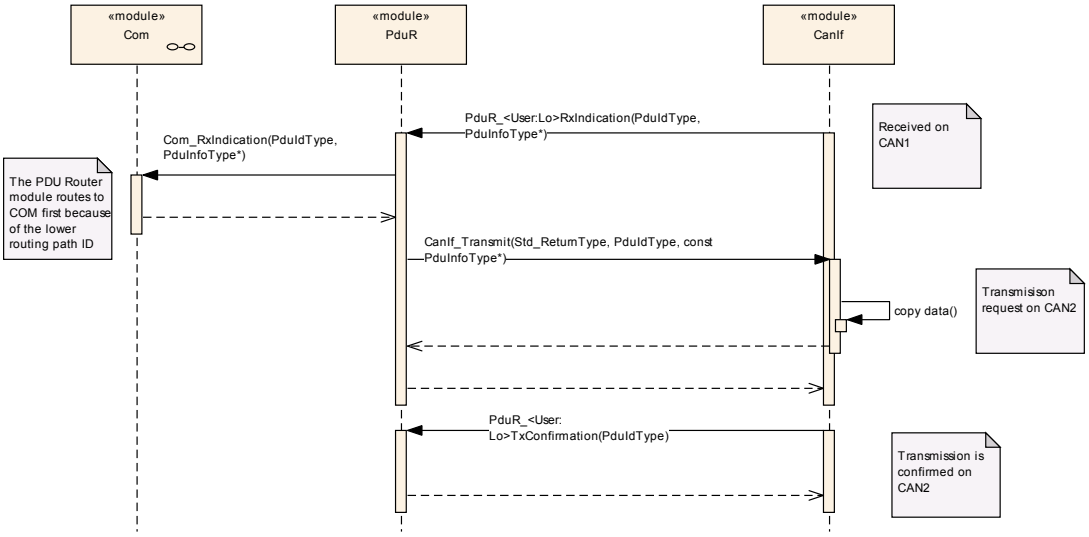

Figure 19: – Gateway of I-PDU from CAN to LIN

9.3.5 Gateway I-PDU using transport protocol modules

Following Figure 20 shows how an (multi N-PDU) I-PDU is gatewayed between two CAN networks, using transport protocol module.

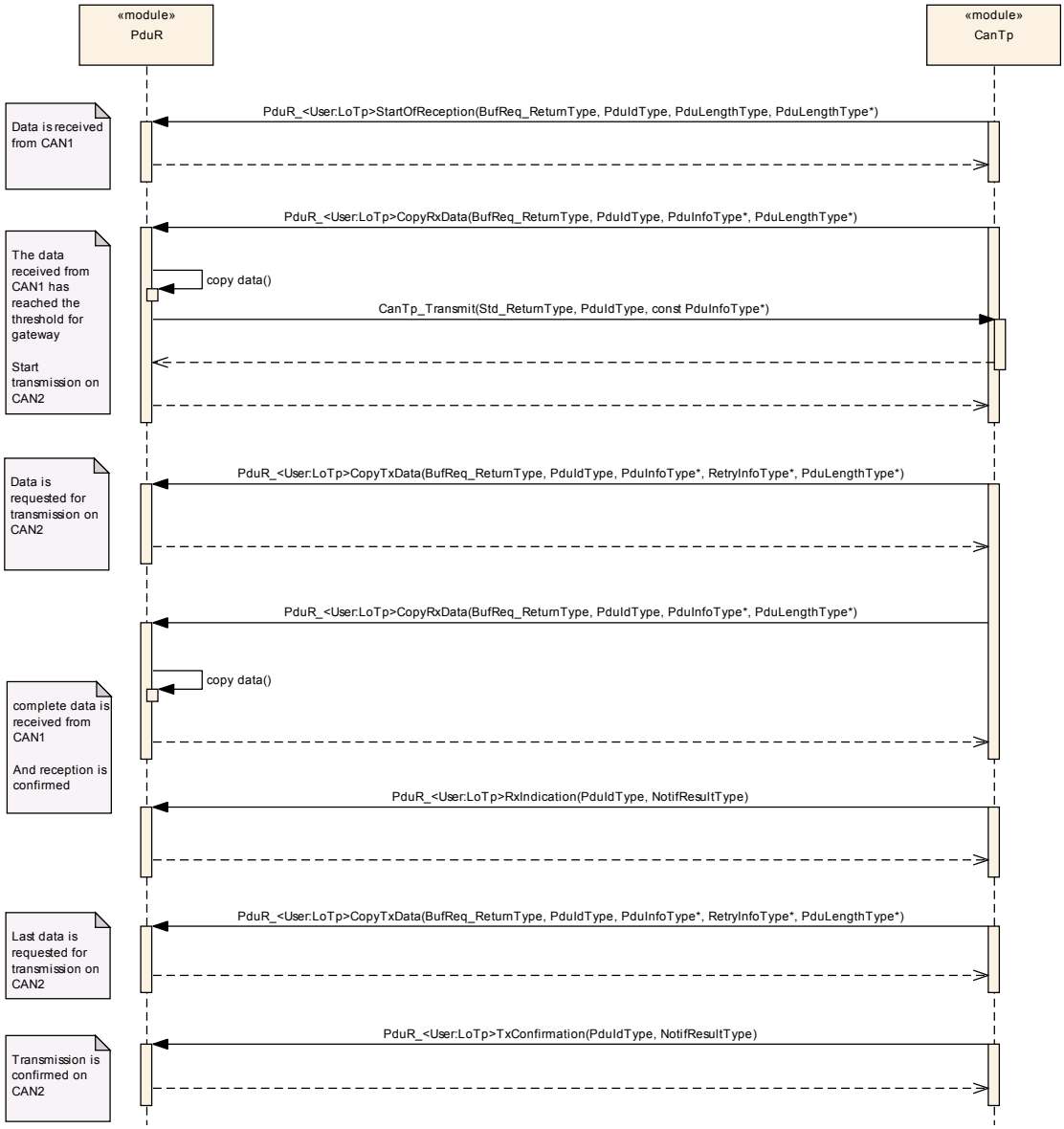

Figure 20: – Gateway of I-PDU (multi N-PDU) between two CAN networks

9.3.6 Gateway I-PDU from CAN1 to DCM and CAN2

The following use-case, Figure 21, shows an I-PDU (contained in a SF) received from CAN1 transport protocol and gatewayed to DCM (internal) and gatewayed to CAN2 transport protocol.

The I-PDU must be buffered in the PDU Router since the DCM module is not aware of that it will be gatewayed to CAN2. Such gatewaying is controlled by the configuration and cannot be processed by the DCM.

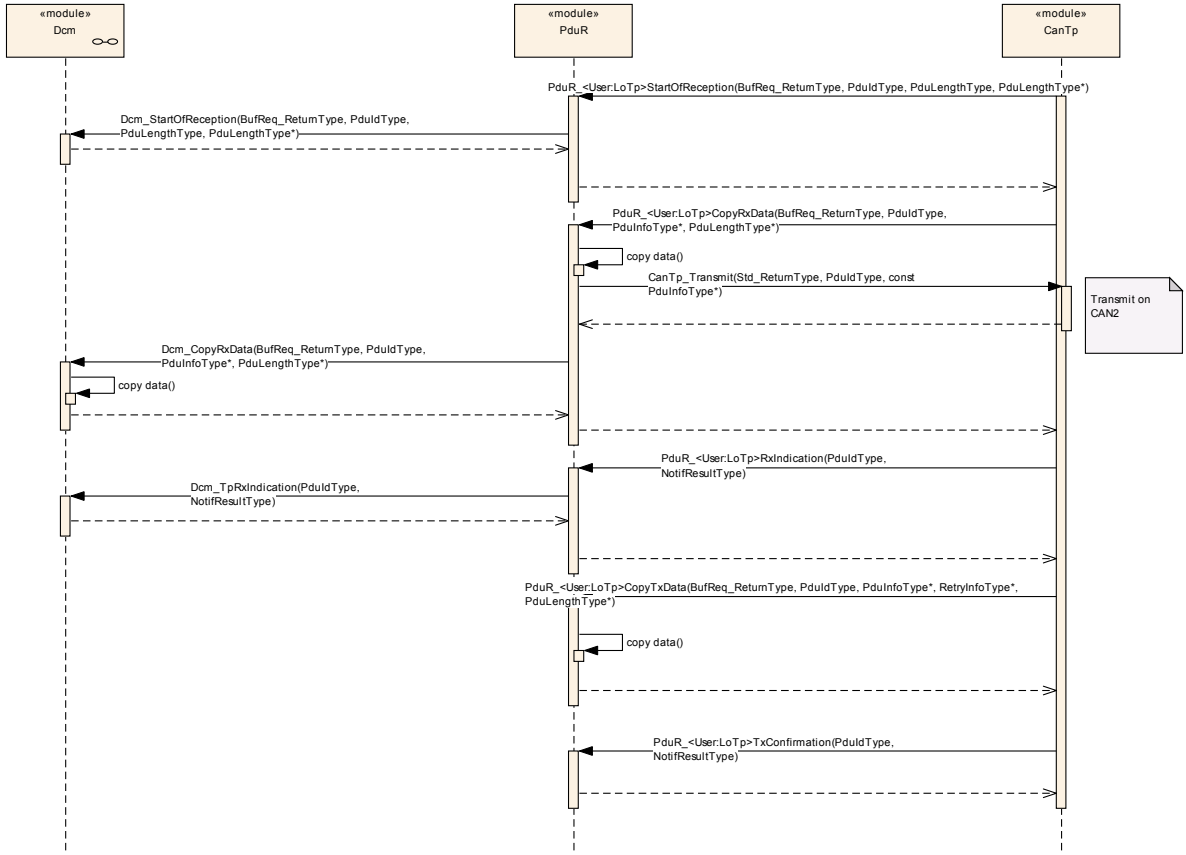

Figure 21: – Gateway of I-PDU to DCM and CAN

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 [7]
  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.

[PDUR295] ⌈The PDU Router module shall support the update of the routing configuration (i.e. the PDU Router routing tables) at post build-time if this variant is supported. ⌋( )

Support of post-build update of the routing table is not always desired. Therefore post-build update of the routing table is only supported in the variant post-build of the PDU Router module, see further section 10.1.2.

The post-build comes in two flavors: Selectable and Loadable, there is no restriction on using any of them in the PDU Router module or even a combination of them.

[PDUR296] ⌈If the variant post-build is supported, the update of the routing tables shall only be possible when the PDU Router module is uninitialized. ⌋( )

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 the API PduR_GetConfigurationId, see section 8.3.1.3.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

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

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

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

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

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

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

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

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

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and supports both Loadable and Selectable flavors.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in chapter 7 and chapter 8. An overview of the top-level PDU Router configuration container PduR is shown in Figure 22.

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. ⌋ (BSW00397)

The zero cost operation is valid only for pre-compile variant.

[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. ⌋ (BSW00399)

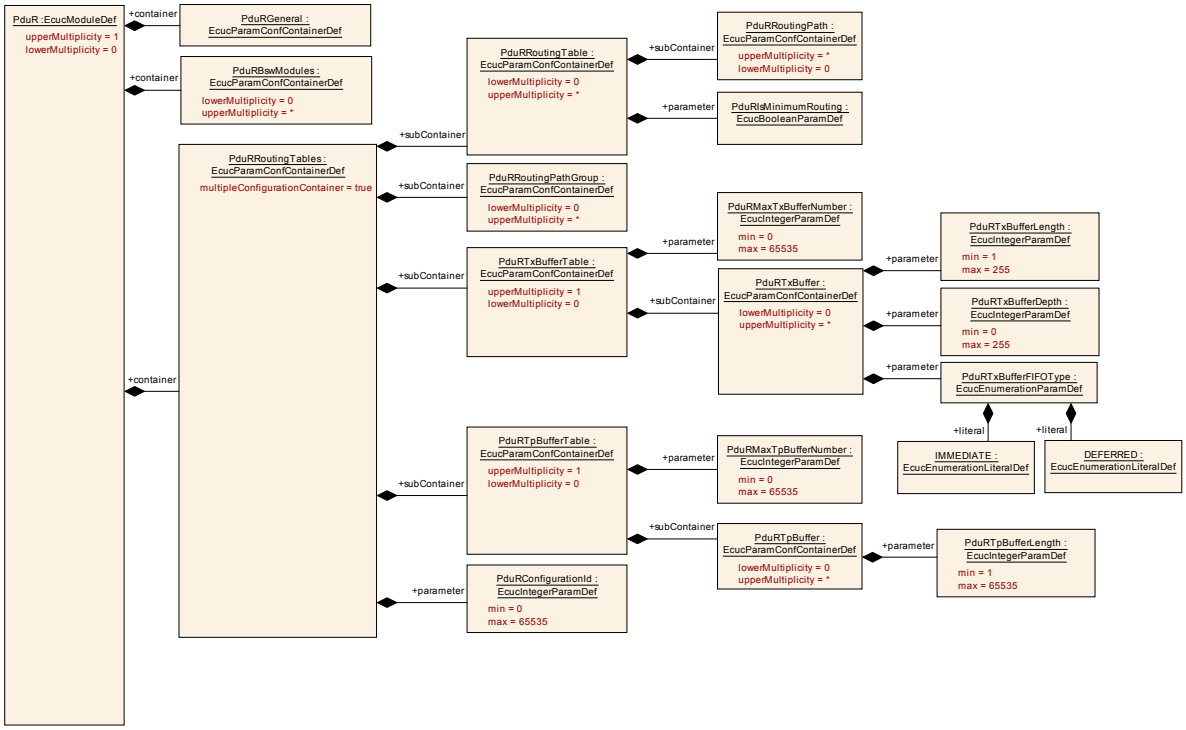

Figure 22: PDU Router Configuration Overview - PduR

10.2.2 PduR

SWS Item	**PDUR293_Conf :**
Module Name	*PduR*
Module Description	Configuration of the PduR (PDU Router) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRBswModules	0..*	Each container describes a specific BSW module (upper/CDD/lower/IpduM) that the PDU Router shall interface to. The reason to have it as own configuration container instead of implication of the routing path is to be able to configure CDD:s properly and to force module's to be used in a post-build situation even though no routing is made to/from this module (future configurations may include these modules).
PduRGeneral	1	This container is a subcontainer of PduR and specifies the general configuration parameters of the PDU Router.
PduRRoutingTables	1	Represents one table of routing paths. This routing table allows multiple configurations that can be used to create several routing tables in the

		same configuration. This is mainly used for post-build (e.g. post-build selectable) but can be used by pre-compile and link-time for variant handling.

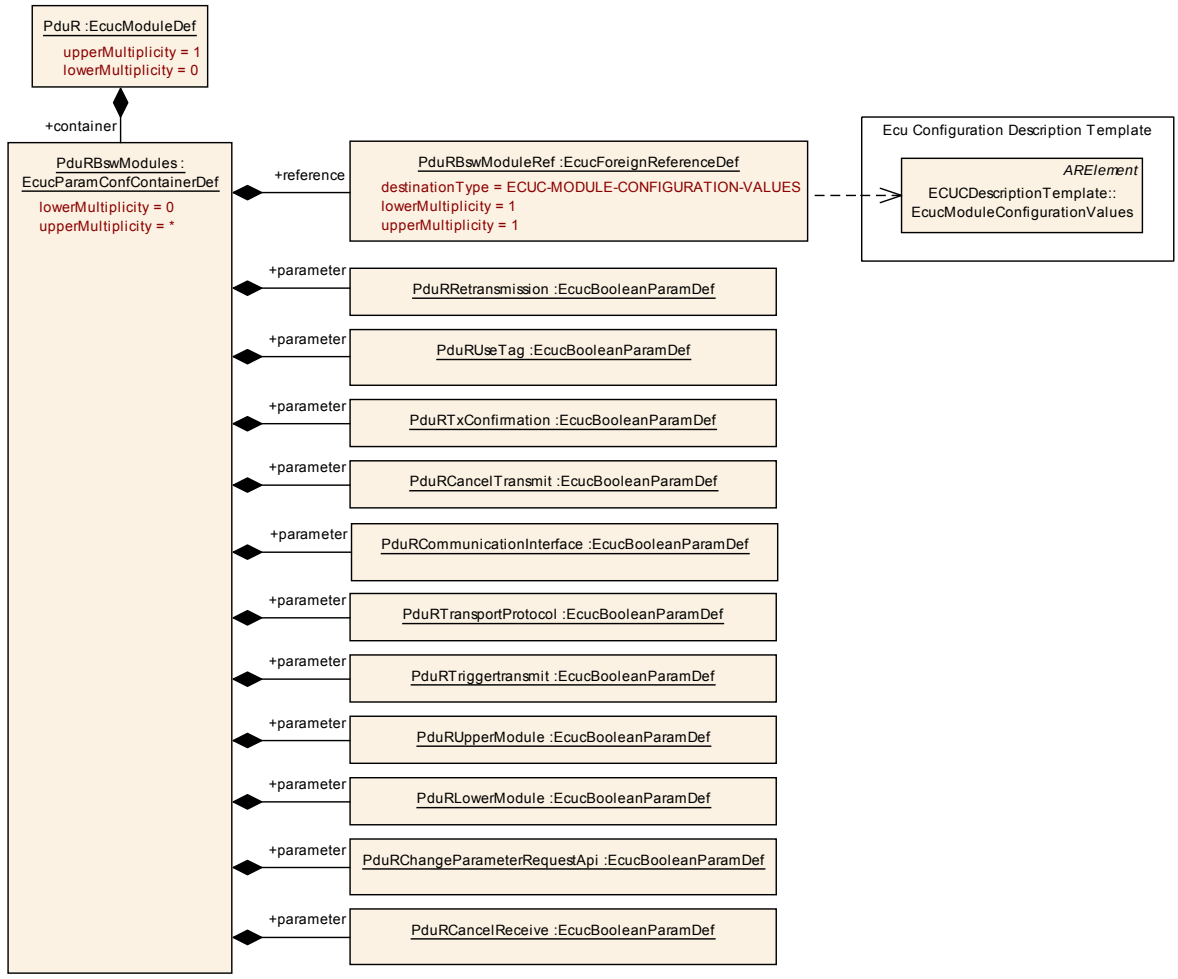

Figure 23 - PduRBswModules

10.2.3 PduRBswModules

SWS Item	**PDUR295_Conf :**
Container Name	PduRBswModules
Description	Each container describes a specific BSW module (upper/CDD/lower/IpduM) that the PDU Router shall interface to. The reason to have it as own configuration container instead of

	implication of the routing path is to be able to configure CDD:s properly and to force module's to be used in a post-build situation even though no routing is made to/from this module (future configurations may include these modules).
Configuration Parameters	

SWS Item	**PDUR340_Conf :**		
Name	PduRCancelReceive		
Description	Specifies if the Transport protocol module supports the CancelReceive API or not. Value true the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR297_Conf :**		
Name	PduRCancelTransmit		
Description	Specifies if the BSW module supports the CancelTransmit API or not. Value true the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR326_Conf :**
Name	PduRChangeParameterRequestApi
Description	This parameter, if set to true, enables the PduR_<Up>ChangeParameterRequest Api for this Module.

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR298_Conf :**		
Name	PduRCommunicationInterface		
Description	Specifies if the BSW module supports the Communication Interface APIs or not. Value true the APIs are supported. A module can have both Communication Interface APIs and Transport Protocol APIs (e.g. the COM module).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR307_Conf :**		
Name	PduRLowerModule		
Description	The PduRLowerModule will decide who will call the APIs and who will implement the APIs. For example, if the CanIf module is referenced then the PDU Router module will implement the PduR_CanIfRxIndication API. And the PDUR module will call the CanIf_Transmit API. Other APIs are of course also covered. An upper module can also be an lower module (e.g. the IpduM module).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR332_Conf :**		
Name	PduRRetransmission		
Description	If set to true this means that the destination transport protocol module will use the retransmission feature. This parameter might be set to false if the retransmission feature is not used, even though the destination transport protocol is supporting it. This parameter is only valid for transport protocol modules and gateway operations. If transmission from a local upper layer module this module will handle the retransmission.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR312_Conf :**		
Name	PduRTransportProtocol		
Description	The PDU Router module shall use the API parameters specified for transport protocol interface.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR313_Conf :**
Name	PduRTriggertransmit

Description	Specifies if the BSW module supports the TriggerTransmit API or not. Value true the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR314_Conf :**		
Name	PduRTxConfirmation		
Description	Specifies if the BSW module supports the TxConfirmation API or not. Value true the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR338_Conf :**
Name	PduRUpperModule
Description	The PduRUpperModule will decide who will call the APIs and who will implement the APIs. For example, if the COM module is referenced then the PDU Router module will implement the PduR_Transmit API. And the PDUR module will call the Com_RxIndication API. Other APIs are of course also covered. An upper module can also be an lower module (e.g. the IpduM module).
Multiplicity	1
Type	EcucBooleanParamDef

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

SWS Item	**PDUR319_Conf :**		
Name	PduRUseTag		
Description	This parameter, if set to true, enables the usage of the tag (<up>) in the following API calls: * PduR_<Up>CancelReceiveRequest * PduR_<Up>CancelTransmitRequest * PduR_<Up>ChangeParameterRequest Example: If used by COM and the parameter is enabled the PduR_ComCancelTransmitRequest is used. The background is that upper layer modules differ in usage of this tag (e.g. COM is using the tag, DCM is not).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR294_Conf :**		
Name	PduRBswModuleRef		
Description	This is a reference to one BSW module's configuration (i.e. not the ECUC parameter definition template). Example, there could be several configurations of LinIf and this reference selects one of them.		
Multiplicity	1		
Type	Foreign reference to [ECUC-MODULE-CONFIGURATION-VALUES]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency			

No Included Containers

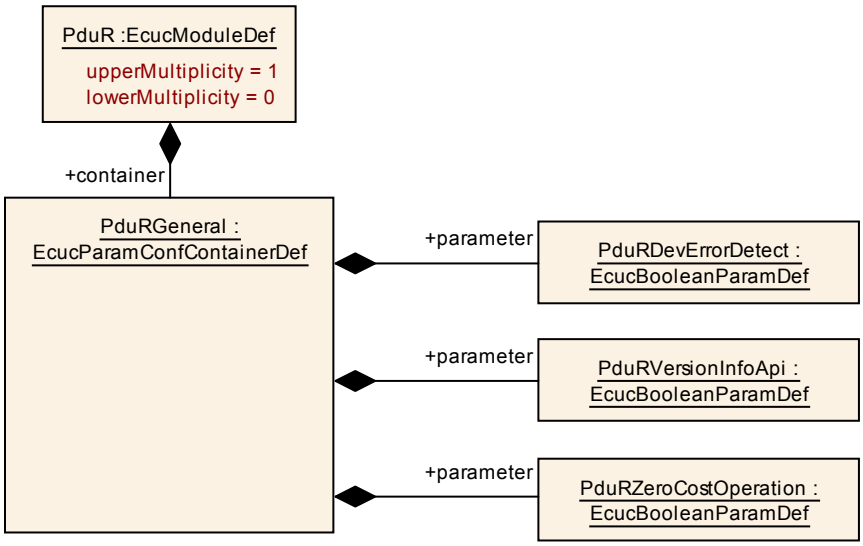

Figure 24 - PduRGeneral

10.2.4 PduRGeneral

SWS Item	**PDUR305_Conf :**
Container Name	PduRGeneral
Description	This container is a subcontainer of PduR and specifies the general configuration parameters of the PDU Router.
Configuration Parameters	

SWS Item	**PDUR302_Conf :**
Name	PduRDevErrorDetect {PDUR_DEV_ERROR_DETECT}
Description	If true then PDU Router will enable the error-reporting to the Development Error Tracer (DET).
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--

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

SWS Item	**PDUR316_Conf :**		
Name	PduRVersionInfoApi {PDUR_VERSION_INFO_API}		
Description	If true the PduR_GetVersionInfo API is available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR317_Conf :**		
Name	PduRZeroCostOperation {PDUR_ZERO_COST_OPERATION}		
Description	If set the PduR configuration generator will report an error if zero-cost-operation cannot be fulfilled. This parameter shall be seen as an input requirement to the configuration generator.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.5 PduRRoutingTables

SWS Item	**PDUR310_Conf :**

Container Name	PduRRoutingTables [Multi Config Container]
Description	Represents one table of routing paths. This routing table allows multiple configurations that can be used to create several routing tables in the same configuration. This is mainly used for post-build (e.g. post-build selectable) but can be used by pre-compile and link-time for variant handling.
Configuration Parameters	

SWS Item	**PDUR327_Conf :**		
Name	PduRConfigurationId		
Description	Identification of the configuration of the PduR configuration. This identification can be read using the PduR API.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRRoutingPathGroup	0..*	This container groups routing path destinations. Destinations are used instead of routing paths since a routing path can be 1:n. It is desirable to be able to enable/disable a specific bus (i.e. a destination) rather than a routing path. Of course it is possible to create groups that covers specific routing paths as well. Enabling and disabling of routing path groups are made using the PduR API
PduRRoutingTable	0..*	Represents one container of routing paths. Each container is either minimum routing or not.
PduRTpBufferTable	0..1	This container will specify the needed buffers for gatewaying using TP. It is not connected to the specific routing path destination to allow a more efficient buffer handling.
PduRTxBufferTable	0..1	This container will specify the needed buffers for gatewaying using

		communication interface. It not defined per routing path to allow reusage of buffers.

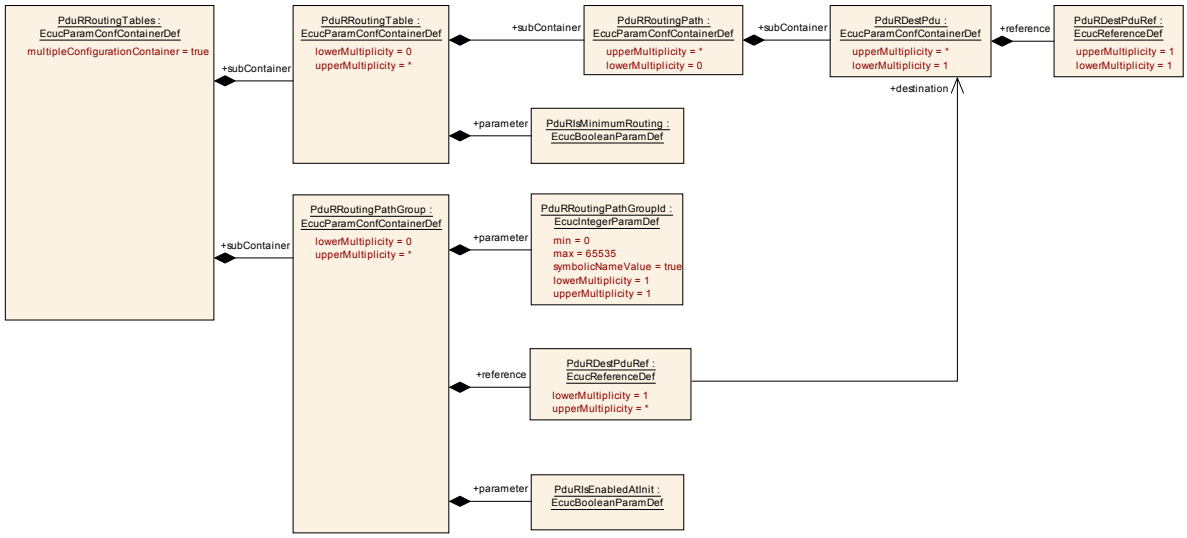

Figure 25 - PduRRoutingPathGroup

10.2.6 PduRRoutingPathGroup

SWS Item	**PDUR308_Conf :**
Container Name	PduRRoutingPathGroup
Description	This container groups routing path destinations. Destinations are used instead of routing paths since a routing path can be 1:n. It is desirable to be able to enable/disable a specific bus (i.e. a destination) rather than a routing path. Of course it is possible to create groups that covers specific routing paths as well. Enabling and disabling of routing path groups are made using the PduR API
Configuration Parameters	

SWS Item	**PDUR329_Conf :**
Name	PduRIsEnabledAtInit
Description	If set to true this routing path group will be enabled after initializing the PDU Router module (i.e. enabled in the PduR_Init function).

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR309_Conf :**		
Name	PduRRoutingPathGroupId		
Description	Identification of the routing group. The identification will be used by the disable/enable API in the PDU Router module API.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**PDUR301_Conf :**		
Name	PduRDestPduRef		
Description	This reference selects one destination of the routing path.		
Multiplicity	1..*		
Type	Reference to [PduRDestPdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.7 PduRRoutingTable

SWS Item	**PDUR247_Conf :**
Container Name	PduRRoutingTable
Description	Represents one container of routing paths. Each container is either minimum routing or not.
Configuration Parameters	

SWS Item	**PDUR306_Conf :**		
Name	PduRIsMinimumRouting		
Description	Specifies if the container contains routing paths that are of the type minimum routing or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

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

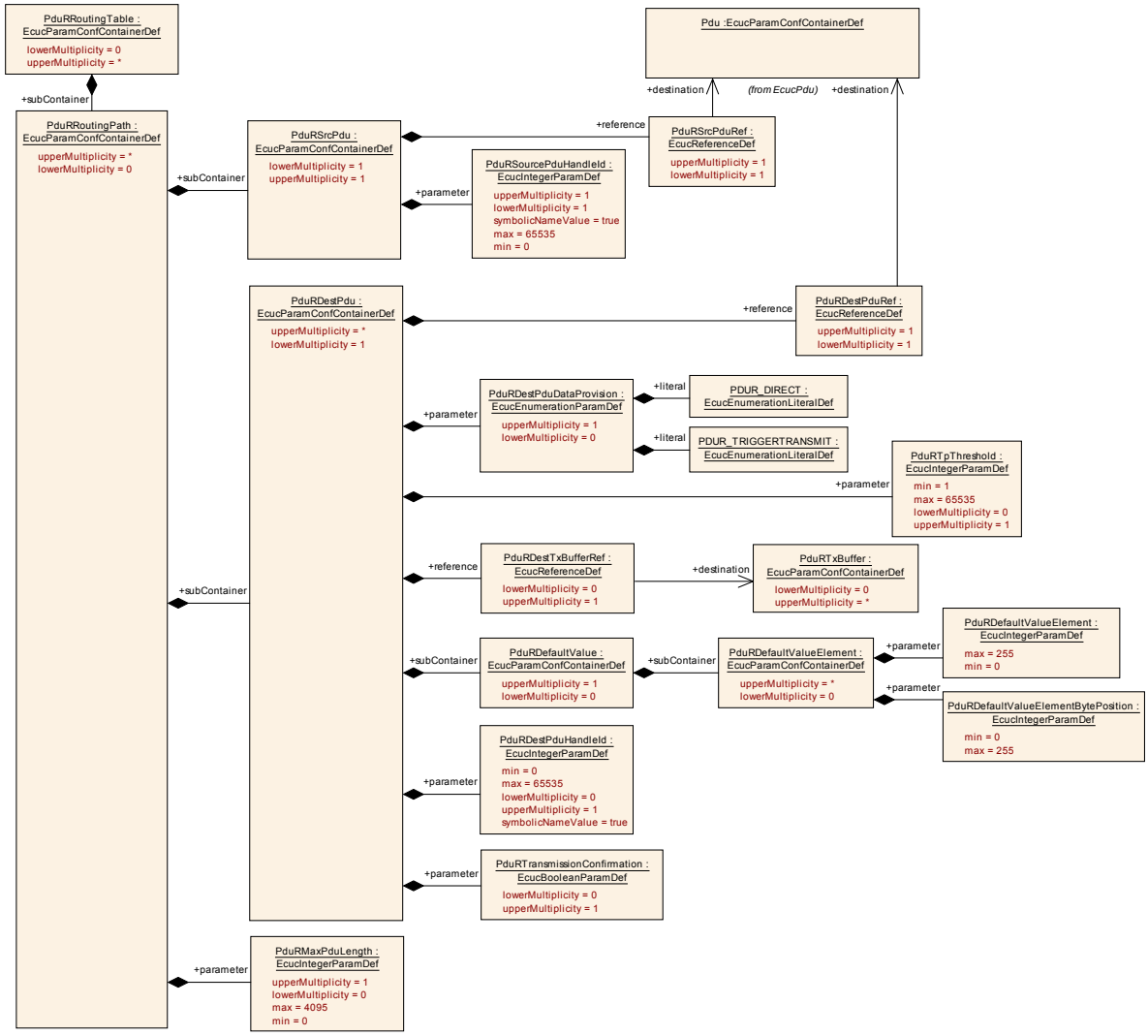

Figure 26 - PduRRoutingPath

10.2.8 PduRRoutingPath

SWS Item	**PDUR248_Conf :**
Container Name	PduRRoutingPath
Description	This container is a subcontainer of PduRRoutingTable and specifies the routing path of a PDU.
Configuration Parameters	

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

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.

10.2.9 PduRDestPdu

SWS Item	**PDUR249_Conf :**
Container Name	PduRDestPdu
Description	This container is a subcontainer of PduRRoutingPath and specifies one destination for the PDU to be routed.
Configuration Parameters	

SWS Item	**PDUR289_Conf :**		
Name	PduRDestPduDataProvision		
Description	Specifies how data are provided: direct (as part of the Transmit call) or via the TriggerTransmit callback function. Only required for non-TP I-PDUs (local and gatewayed).		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	PDUR_DIRECT		The PDU Router module shall call the transmit function in the destination module and not buffer the I-PDU
	PDUR_TRIGGERTRANSMIT		The PDU Router module shall call the transmit function in the destination module. The destination module will request the I-PDU using the triggerTransmit function. The I-PDU is shall be buffered.
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-

			COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**PDUR322_Conf :**		
Name	PduRDestPduHandleId		
Description	PDU identifier assigned by PDU Router. Used by communication interface and transport protocol modules for confirmation.		
Multiplicity	0..1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**PDUR320_Conf :**		
Name	PduRTpThreshold		
Description	Defines the number of bytes which shall be received before transmission on the destination bus may start. Only required for routing-on-the-fly TP gateway PDUs. The threshold shall not be larger than the length of the related TP Buffer.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR339_Conf :**		
Name	PduRTransmissionConfirmation {PDUR_TRANSMISSION_CONFIRMATION}		
Description	This parameter is only for communication interfaces. Transport protocol modules will always call the TxConfirmation function. If set the destination communication interface module will call the TxConfirmation. However the TxConfirmation may be not called due to error. So the PduR shall not block until the TxConfirmation is called. One background for this parameter is for the PduR to know when all modules have confirmed a multicast operation.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR291_Conf :**		
Name	PduRDestPduRef		
Description	Destination PDU reference; reference to unique PDU identifier which shall be used by the PDU Router instead of the source PDU ID when calling the related function of the destination module.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**PDUR304_Conf :**
Name	PduRDestTxBufferRef
Description	Reference to a buffer that is allocated in the PduRTxBuffer. Having a global (for PduR) list of

	buffers allows reusage and hence less memory consumption.		
Multiplicity	0..1		
Type	Reference to [PduRTxBuffer]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRDefaultValue	0..1	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.

10.2.10 PduRSrcPdu

SWS Item	**PDUR288_Conf :**
Container Name	PduRSrcPdu
Description	This container is a subcontainer of PduRRoutingPath and specifies the source of the PDU to be routed.
Configuration Parameters	

SWS Item	**PDUR311_Conf :**		
Name	PduRSourcePduHandleId		
Description	PDU identifier assigned by PDU Router.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**PDUR318_Conf :**		
Name	PduRSrcPduRef		
Description	Source PDU reference; reference to unique PDU identifier which shall be used for the requested PDU Router operation.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.11 PduRDefaultValue

SWS Item	**PDUR299_Conf :**
Container Name	PduRDefaultValue
Description	Specifies the default value of the I-PDU. Only required for gateway operation and if at least one PDU specified by PduRDestPdu uses TriggerTransmit Data provision. Represented as an array of IntegerParamDef.
Configuration Parameters	

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

10.2.12 PduRDefaultValueElement

SWS Item	**PDUR300_Conf :**
Container Name	PduRDefaultValueElement
Description	Each value element is represented by the element and the position in an array.
Configuration Parameters	

SWS Item	**PDUR290_Conf :**		
Name	PduRDefaultValueElement		
Description	The default value consists of a number of elements. Each element is one byte long and the number of elements is specified by SduLength. The position of this parameter in the container is specified by the PduRElementBytePosition parameter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**PDUR292_Conf :**		
Name	PduRDefaultValueElementBytePosition		
Description	This parameter specifies the byte position of the element within the default value		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.13 PduRTpBufferTable

SWS Item	**PDUR335_Conf :**
Container Name	PduRTpBufferTable
Description	This container will specify the needed buffers for gatewaying using TP. It is not connected to the specific routing path destination to allow a more efficient buffer handling.
Configuration Parameters	

SWS Item	**PDUR330_Conf :**		
Name	PduRMaxTpBufferNumber		
Description	maximum number of TP buffers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

Included Containers		
Container Name	*Multiplicity*	*Scope / Dependency*
PduRTpBuffer	0..*	Specifies a buffer used for gatwaying through TP.

10.2.14 PduRTpBuffer

SWS Item	**PDUR334_Conf :**

Container Name	PduRTpBuffer
Description	Specifies a buffer used for gatwaying through TP.
Configuration Parameters	

SWS Item	**PDUR325_Conf :**		
Name	PduRTpBufferLength		
Description	Length of the TP buffer in number of bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.15 PduRTxBufferTable

SWS Item	**PDUR337_Conf :**
Container Name	PduRTxBufferTable
Description	This container will specify the needed buffers for gatewaying using communication interface. It not defined per routing path to allow reusage of buffers.
Configuration Parameters	

SWS Item	**PDUR331_Conf :**	
Name	PduRMaxTxBufferNumber	
Description	maximum number of Tx buffers.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	--	

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PduRTxBuffer	0..*	Specifies a buffer used for gatwaying through communication interface.

10.2.16 PduRTxBuffer

SWS Item	**PDUR336_Conf :**
Container Name	PduRTxBuffer
Description	Specifies a buffer used for gatwaying through communication interface.
Configuration Parameters	

SWS Item	**PDUR324_Conf :**		
Name	PduRPduMaxLength		
Description	Length of the Tx buffer in number of bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**PDUR323_Conf :**
Name	PduRTxBufferDepth
Description	Number of Pdus that can be stored in the buffer. If

	value is 1 then the buffer semantic is "last is best". If the value is greater then 1 then the buffer semnatic is a FiFo.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3 Published Information

[PDUR0778] ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [10].⌋( )

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

11 PDU Router module design notes

This chapter collects a set of notes that describes features of this document.

11.1 Backwards compatibility

The APIs for the Transport Protocol modules was redesigned in AUTOSAR release 4.0. Therefore the APIs are not directly backwards compatible with AUTOSAR release 3.1 and older releases. To be able to use modules from different AUTOSAR releases then a compatible layer is needed. This chapter will describe how such layer may be designed.

The basic idea of the new TP API is that the copying of data is moved from the TP modules to the PDU Router module (in case of gateway) or upper layer receiver/transmitter module.

The background for the change was to solve two issues:

- Gatewaying-on-the-fly could not be implemented efficiently.
- The retransmission needed by specific busses (e.g. FlexRay).

Following to sections describe a possible solution for using either an AUTOSAR 3.1 (or older) upper layer, or an AUTOSAR 3.1 (or older) lower layer.

The upper layer module may be a PDU Router module, COM, DCM or other upper layer module that operates the TP API.

11.1.1 AUTOSAR 4.0 upper layer and AUTOSAR 3.1 (or older) TP

Following two figures shows where an AUTOSAR 4.0 upper layer is used together with a lower layer 3.1 TP.

The Figure 27 shows a use-case where the upper layer is receiving the I-PDU.

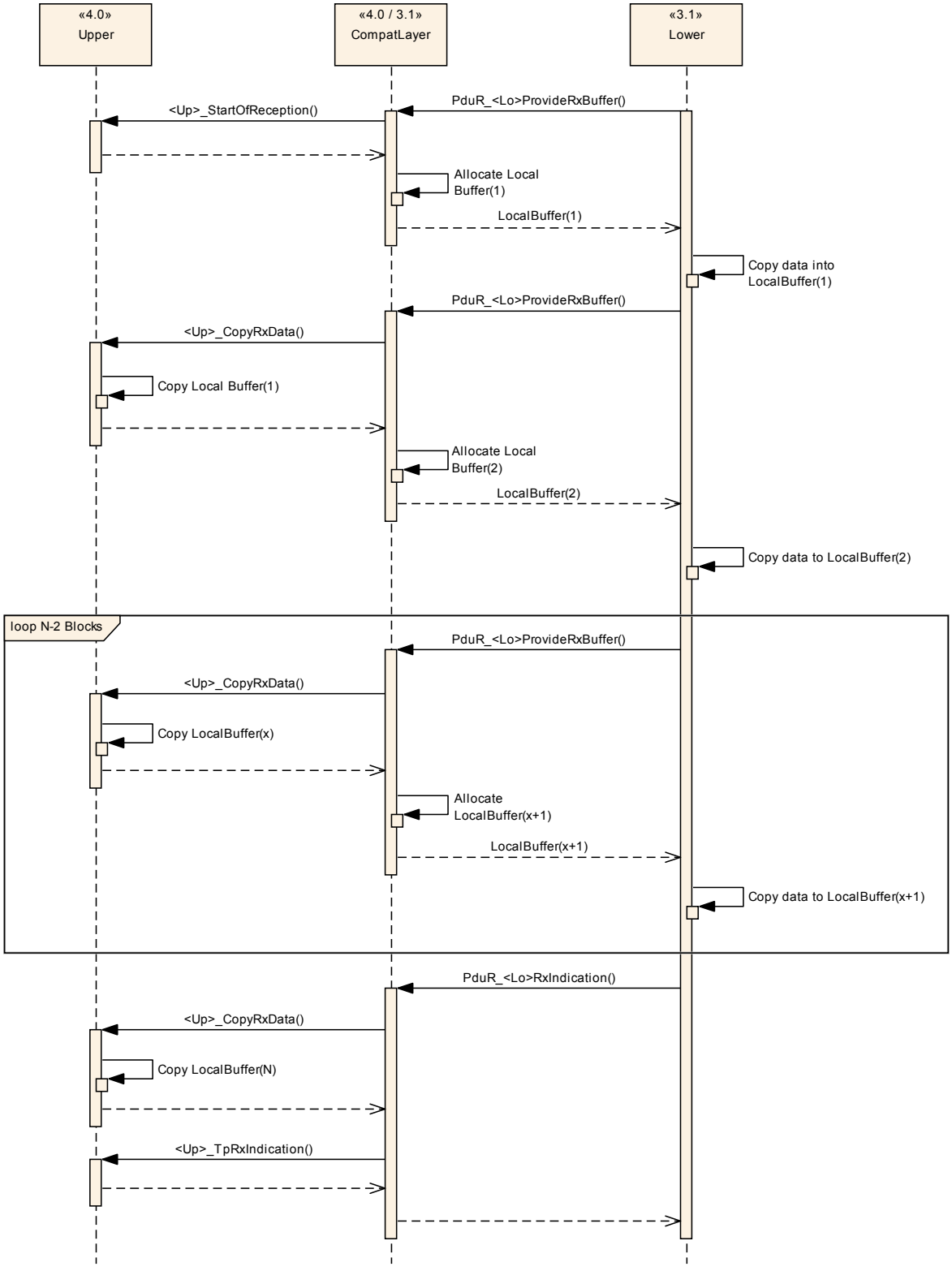

Figure 27 - Reception-Lower3.1-Upper4.0

The Figure 28 shows a use-case where the upper layer is transmitting the I-PDU.

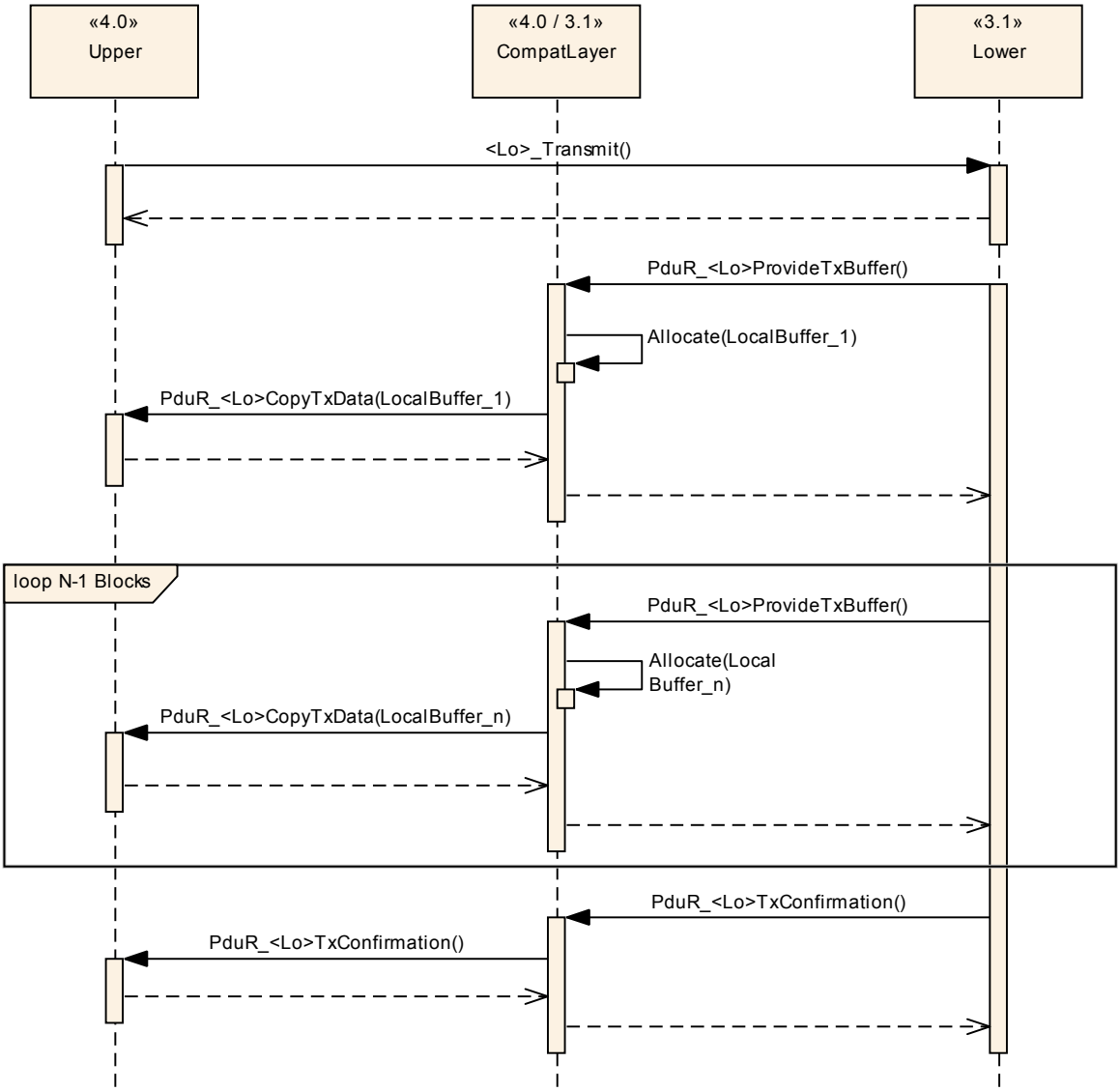

Figure 28 - Transmission-Upper4.0-Lower3.1

11.1.2 AUTOSAR 3.1 (or older) TP and AUTOSAR 4.0 upper layer

Following two figures show where an AUTOSAR 3.1 upper layer is used together with a lower layer 4.0 TP.

The Figure 29 shows a use-case where the upper layer is receiving the I-PDU.

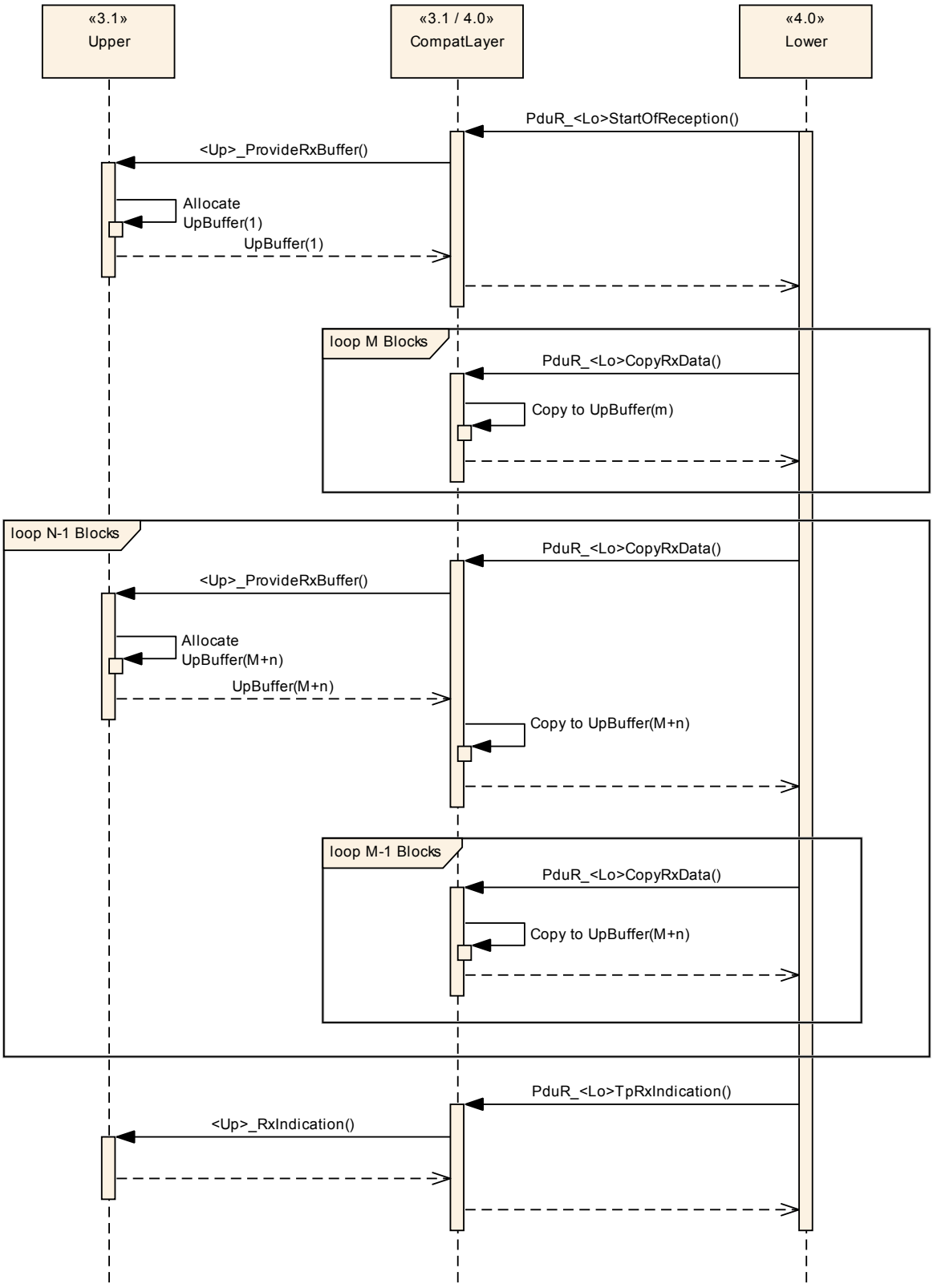

Figure 29 - Reception-Lower4.0-Upper3.1

The Figure 30 shows a use-case where the upper layer is transmitting the I-PDU.

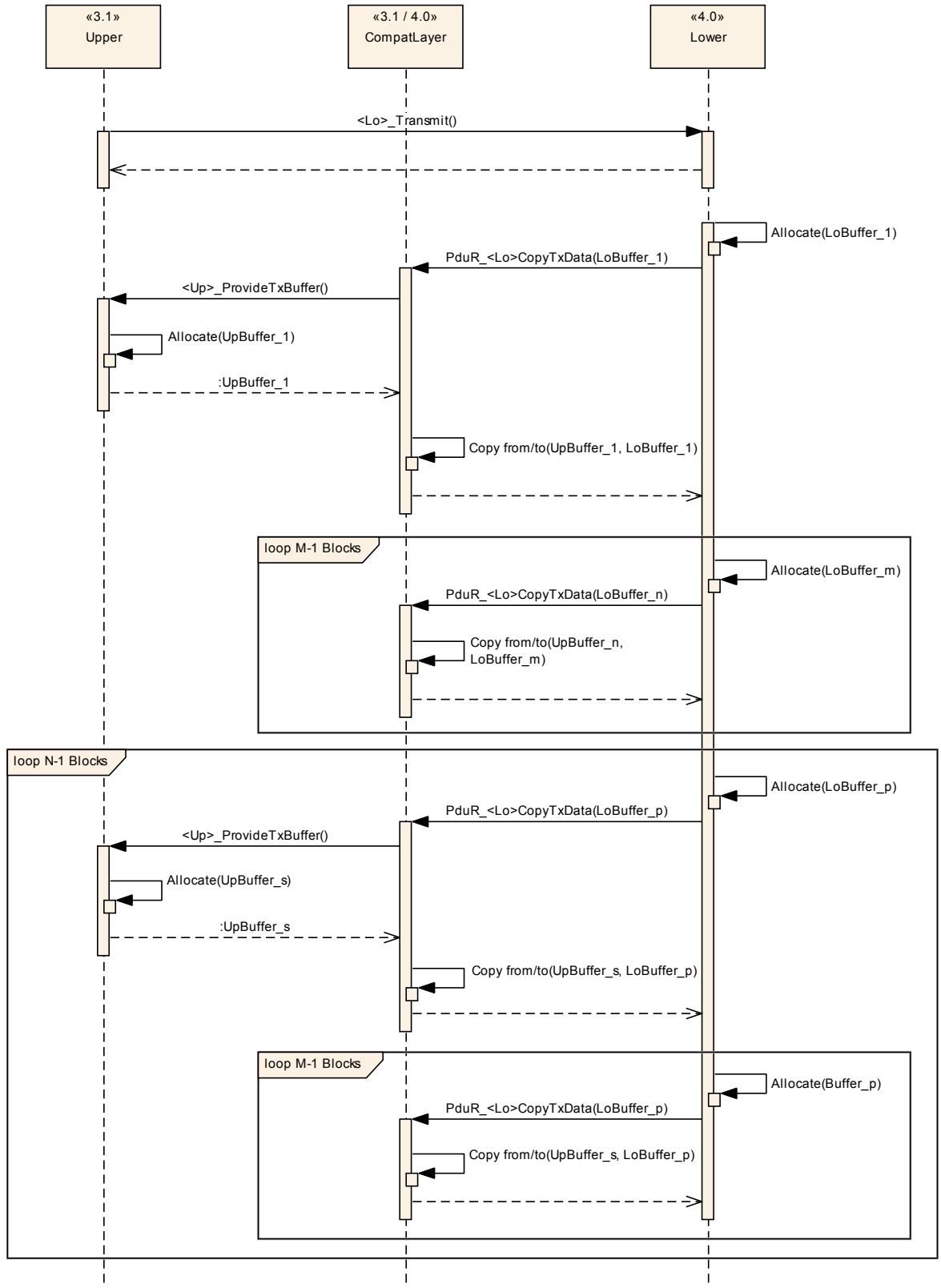

Figure 30 - Transmission-Upper3.1-Lower4.0

11.2 Configuration parameter considerations

The configuration parameters listed in chapter 10 contain restrictions for the parameters themselves. But no restrictions are set that affects more than one parameter. The intention of this section is to list some of these to better understand the configuration parameters and also to allow a simpler configuration generator tool for the PDU Router module.

The buffers needed for gatewaying (communication interface and transport protocol) are specified per destination in the configuration. Since no specific traffic shaping can be specified it is assumed that worst case where all I-PDUs are gatewayed at the same time. It is possible to extend the configuration with parameters that allow more efficient usage of buffers.

11.3 Generic interfaces concept

The provided and used APIs of the PDU Router module are not connected to specific busses. The API names in chapter 8.3 have a generic part (<Up>, <Lo>, etc) that will be exchanged with the name of the module using or implementing the API.

The way to identify the name is using the reference to an ECUC description, see Figure 31. The short-name will be used in the referenced ECUC description.

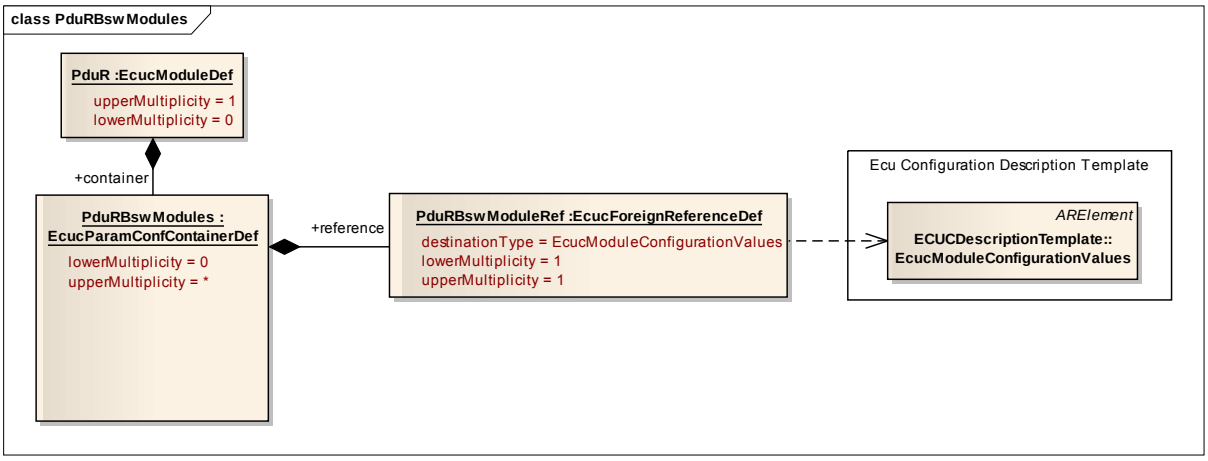

Figure 31 - Generic interface configuration

The PduRBswModules container contain parameters that describe the supported functionality (if it is communication interface, transport layer, upper layer, lower layer,

etc.) of the BSW module.

The connection between the generic interface configuration of a BSW module and the I-PDUs are made using the routing paths and the I-PDU configuration in the ECUC module.

11.4 Example structure of Routing tables

This chapter shows example structures of routing tables that contain the properties of each I-PDU. It does not specify the internals of the PDU Router but shall rather serve as example for better understanding of APIs and I-PDUs.

The IpduM is not considered by these examples.

Note: This chapter is by no means the recommended implementation way. The chapter focuses more on understandability than optimizing implementation.

11.4.1 Transmission and multicast via communication interface modules

Routing table used by PduR_ComTransmit for I-PDUs transmitted by COM:

PduR_ComTransmit (PduIdType id, PduInfoType* info)			
id	***TargetFctPtr***	***TargetPduId***	***Description***
0	CanIf_Tansmit	0	Transmission on CanIf
1	FrIf_Transmit	0	Transmission on FrIf
2	CanIf_Tansmit	1	Transmission on CanIf
3	CanIf_Transmit CanIf_Transmit	0 2	Multicast using CanIf on two CAN busses
4	LinIf_Transmit CanIf_Transmit	2 3	Multicast using CanIf and LinIf. Note that for LinIf this is a sporadic frame (will later be a TriggerTransmit call).

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 module to two different CAN busses and COM module to LinIf

and CanIf. For the latter an internal PDU Router function (MultiIf_Transmit) and an additional routing table is used.

The destination module will confirm the transmission of the I-PDU using the configured I-PDU id, and it might not be the same as in the transmit call.

11.4.2 Reception and gateway via communication interface modules

Routing table used by PduR_<Lo>RxIndication for receiving I-PDUs received from the lower layer communication interfaces:

PduR_<Lo>RxIndication (PduIdType id, PduInfoType* info)			
id	***TargetFctPtr1***	***TargetPduId***	***Description***
0	Com_RxIndication	0	Routed to COM module
1	Com_RxIndication CanIf_Transmit	0 1	Routed to COM and gatewayed to CanIf
2	CanIf_Transmit LIN	1 2	Gatewayed to CanIf and to LinIf. In the LinIf case the LinIf will later call TriggerTransmit. The PDU Router ill not call LinIf_Transmit

11.5 Configuration generator

The PDU Router configuration generator will take the ECU configuration description XML file containing the PDU Router configuration as input. And the generator will produce .c and .h files containing the configuration.

One aim of the configuration generator is to allow the generator to produced an efficient PDU Router module implementation. Since the PDU Router module is a central module it is important that the final executable including configuration is efficient as possible:

[PDUR0764] ⌈The PDU Router module generator shall be able to optimize away features based on if they are used or not. At least following features shall be considered:

- Transport protocol
- Communication interfaces
- Gateway
- FIFO queue handling

One part of the job made by the generator is to lookup all routing paths and produces the correct look-up tables and the correct APIs to be used. Here are some examples how the generator may handle the routing paths. ⌋(BSW06020)

11.5.1 CanIf and COM routing path example

This is an example that shows how an I-PDU received by the CanIf module and forwarded by the COM module is handled.

In Figure 32 the configuration of CanId, COM and PDU Router is shown. The PDU Router has a routing path with a source I-PDU (SrcPduRef) and destination I-PDU (DestPduRef). When following the I-PDU SrcPduRef it is found that the CanIf PduIdRef is pointing at the same I-PDU in the ECUC. The DestPduRef is followed and it is found that COM PduIdRef is pointing at the same I-PDU in the ECUC.

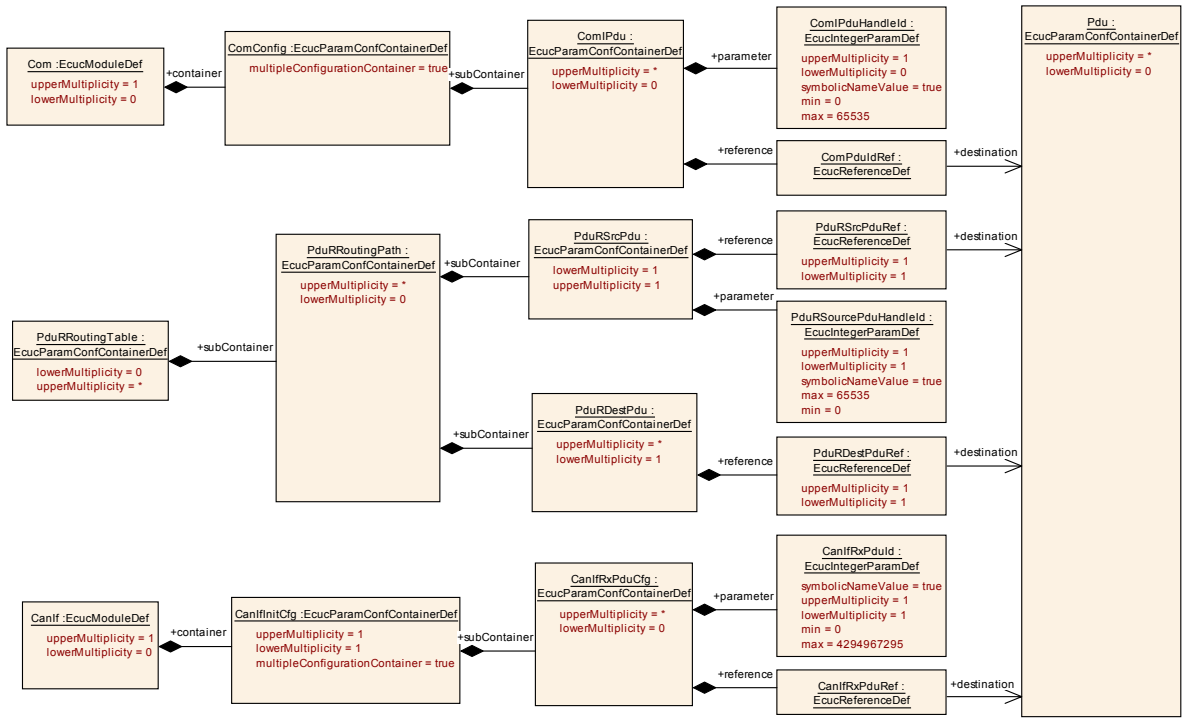

Figure 32: PDU Router, CanIf and COM configuration example

The CanIfCanRxPduId reveals the I-PDU ID for the source I-PDU and the ComIPduHandleId reveals the I-PDU ID for the destination I-PDU.

The shortname of the CanIf module and the COM module (and that the I-PDU is transported on a communication interface module) will generate the routing table and APIs to be used:

PduR_<user>IfRxIndication (PduIdType id, PduInfoType* info)			
id	***Source***	***TargetPduId***	***Destination***
12	CanIf_RxIndication	13	Com_RxIndication

PduR_COM.h
void PduR_ComRxIndication(PduIdType id, PduInfoType* info);

PduR_CanIf.h
void PduR_CanIfRxIndication(PduIdType id, PduInfoType* info);

If the zero-cost-operation is enabled and the CanId module is only forwarding (through PDU Router module) to the COM module, the generator may generate the optimization (if source code is used):

#define PduR_CanIfRxIndication PduR_ComRxIndication

11.6 Post-build considerations

This section describes some important behavior when using the post-build variant of the PDU Router. It contains no requirements, just important issues that need to be considered.

NVRAM and RAM memory size can potentially grow if a new post-build configuration is downloaded into the ECU. Estimation at design time must be done to allow such grow so other areas are not overwritten (in case of RAM) or memory borders are not crossed.

It is not possible to configure restrictions/locations/etc of memory in the PDUR module configuration since this is implementation specific and relitevly difficult to implement (pre-compile and link-time does not really need this). It is recommended for implementations of PDUR module generators to extend the configuration with specific memory constraints if needed.

12 Generic COM services

The PDU router module is modeled as a generic module that can interface to different upper and lower modules. The approach taken to model this generic approach is to have a virtual module called GenericComServices. This virtual module contains a set of APIs that the PDU router will call in upper layer or lower layer modules. Thes APIs are generic in the way that they contain a tag <Lo>, <Up> and <LoTp> that is replaced with the interfaced module. The tag is set by the configuration in the PduRBSWModules container using the PduRBswModuleRef reference parameter.

The following subchapters describe the APIs that are realized by this virtual module GenericComServices.

No requirement tags are put on the interfaces since they are implemented in other BSW modules than the PDU router module.

12.1 Imported types

The GenericComServices will use the following types:

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
	NotifResultType
	PduIdType
	PduInfoType
	PduLengthType
	RetryInfoType
	TPParameterType
Std_Types	Std_ReturnType

12.2 Interfaces implemented in upper layer modules

12.2.1 <Up>_CopyRxData

Service name:	<Provider:UpTp>_CopyRxData	

Syntax:	`BufReq_ReturnType GenericComServices_CopyRxData(`
	`  PduIdType id,`
	`  PduInfoType* info,`
	`  PduLengthType* length`
	`)`	

Service ID[hex]:	0x32	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	--
	info	--
Parameters (inout):	None	
Parameters (out):	length	--
Return value:	BufReq_ReturnType	--
Description:	This function is called when transport protocol module have data to copy to the receiving module. Several calls may be made during one transportation of an I-PDU.	

12.2.2 <Up>_CopyTxData

Service name:	<Provider:UpTp>_CopyTxData	

Syntax:	`BufReq_ReturnType GenericComServices_CopyTxData(`
	`  PduIdType id,`
	`  PduInfoType* info,`
	`  RetryInfoType* retry,`
	`  PduLengthType* length`
	`)`	

Service ID[hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	--
	info	--

	retry	--
Parameters (inout):	None	
Parameters (out):	length	--
Return value:	BufReq_ReturnType	--
Description:	This function is called by the transport protocol module to query the transmit data of an I-PDU segment. Each call to this function copies the next part of the transmit data until TpDataState indicates TP_DATARETRY. In this case the API restarts to copy the data beginning at the location indicated by TpDataCnt.	

12.2.3 <Up>_RxIndication

Service name:	GenericComServices_RxIndication	

Syntax:	`void GenericComServices_RxIndication(`
	`  PduIdType RxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`	

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

12.2.4 <Up>_StartOfReception

Service name:	<Provider:UpTp>_StartOfReception
Syntax:	`BufReq_ReturnType GenericComServices_StartOfReception(`

	PduIdType id, PduLengthType TpSduLength, PduLengthType* length)	
Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	--
	TpSduLength	--
Parameters (inout):	None	
Parameters (out):	length	--
Return value:	BufReq_ReturnType	--
Description:	This function will be called by the transport protocol module at the start of receiving an I-PDU. The I-PDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdulength equal to 0.	

12.2.5 <Up>_TpRxIndication

Service name:	<Provider:UpTp>_TpRxIndication	
Syntax:	void GenericComServices_TpRxIndication(PduIdType id, NotifResultType result)	
Service ID[hex]:	0x33	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	--
	result	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	

Description:	Called by the transport protocol module after an I-PDU has been received successfully or when an error occurred. It is also used to confirm cancellation of an I-PDU.

12.2.6 <Up>_TpTxConfirmation

Service name:	<Provider:UpTp>_TpTxConfirmation	

Syntax:	`void GenericComServices_TpTxConfirmation(`
	`  PduIdType id,`
	`  NotifResultType result`
	`)`	

Service ID[hex]:	0x37	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	--
	result	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by a transport protocol module after the I-PDU has been transmitted on its network, the result will reveal if the transmission was successful or not.	

12.2.7 <Up>_TriggerTransmit

Service name:	GenericComServices_TriggerTransmit

Syntax:	`Std_ReturnType GenericComServices_TriggerTransmit(`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`

Service ID[hex]:	0x41
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.

Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

12.2.8 <Up>_TxConfirmation

Service name:	GenericComServices_TxConfirmation	

Syntax:	`void GenericComServices_TxConfirmation(`
	`  PduIdType TxPduId`
	`)`	

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

12.3 Interfaces implemented in lower layer modules

Following interfaces are interfaces that may be implemented by the lower layer communication interface and transport protocol modules. The PDU router will,

dependent on its configuration, use these interfaces.

12.3.1 <Lo>_Transmit

Service name:	<Provider:Lo>_Transmit	

Syntax:	`Std_ReturnType GenericComServices_Transmit(`
	`  PduIdType TxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TxPduId	--
	PduInfoPtr	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	Requests the sending of a PDU.	

12.4 Interfaces implemented in lower layer communication interface modules

Following interfaces are interfaces that may be implemented by the lower layer communication interface modules. The PDU router will, dependent on its configuration, use these interfaces.

12.4.1 <Lo>_CancelTransmit

Service name:	<Provider:Lo>_CancelTransmit
Syntax:	`Std_ReturnType GenericComServices_CancelTransmit(` `  PduIdType id`

	)	
Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	id	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	Requests cancellation of a specific I-PDU in a lower layer communication interface module.	

12.5 Interfaces implemented in lower layer transport layer modules

Following interfaces are interfaces that may be implemented by the lower layer transport protocol modules. The PDU router will, dependent on its configuration, use these interfaces.

12.5.1 <LoTp>_CancelTransmit

Service name:	<Provider:LoTp>CancelTransmit	
Syntax:	`Std_ReturnType GenericComServicesCancelTransmit( PduIdType id )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	id	Identifiaction of the I-PDU to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request accepted (but not yet performed). E_NOT_OK: request not accepted (e.g. cancellation not possible).

Description:	Requests cancellation of a specific I-PDU in a lower layer transport protocol module.

12.5.2 <LoTp>_CancelReceive

Service name:	<Provider:LoTp>CancelReceive	

Syntax:	`Std_ReturnType GenericComServicesCancelReceive(`
	`  PduIdType id`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	id	Identifiaction of the I-PDU to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request accepted (but not yet performed). E_NOT_OK: request not accepted (e.g. cancellation not possible).
Description:	Requests cancellation of an I-PDU reception in a lower layer transport protocol module.	

12.5.3 <LoTp>_ChangeParameter

Service name:	<Provider:LoTp>_ChangeParameter	

Syntax:	`Std_ReturnType GenericComServices_ChangeParameter(`
	`  PduIdType id,`
	`  TPParameterType parameter,`
	`  uint16 value`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	id	Identifiaction of the I-PDU which the parameter change request

		shall affect.
	parameter	The selected parameter that the request shall change.
	value	The new value of the parameter
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request is accepted. E_NOT_OK: request is not accepted.
Description:	Request to change a specific transport protocol parameter (e.g. block-size).	

13 Changes to Release 3.1

13.1 Deleted SWS Items

SWS Item	*Rationale*
PDUR321	
PDUR322	
PDUR323	
PDUR165	
PDUR426	Variant 2
PDUR224	
PDUR478	This is a design note and not a requirement
PDUR252	Obvious that the integrator shall configure the FIFO
PDUR253	Implementation
PDUR310	Implementation
PDUR312	Implementation
PDUR258	Implementation
PDUR259	Implementation
PDUR0681	The same API name for transmit cancellation is used for Tp and non-Tp routing paths
PDUR484	ChangeParameter API is synchronous, a confirmation is no longer needed
PDUR0735	ChangeParameter API is synchronous, a confirmation is no longer needed
PDUR0770	The return value E_PENDING is removed; PduR_ReturnType is no longer needed
PDUR0712	Restricted multicast Tp transmission to single frames.
PDUR0713	Restricted multicast Tp transmission to single frames.
PDUR341_Conf, PDUR342_Conf, PDUR343_Conf	Production errors PDUR_E_PDU_INSTANCE_LOST and PDUR_E_INIT_FAILED changed to development errors.
PDUR0768	Changed from a requirement to a configuration note because it is a requirement to a user, not to the implementation of the PduR
PDUR305, PDUR0663, PDUR0664, PDUR0662, PDUR0668	Reworked for RfC #50682 (FIFO Handling, no Immediate FIFO anymore

13.2 Replaced SWS Items

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

13.3 Changed SWS Items

SWS Item	*Rationale*
PDUR167	Clarification
PDUR171	Clarification
PDUR242	Correction (BooleanParamDef instead of StringParamDef)
PDUR255	FIFO of size 1 shall be considered as a single buffer
PDUR258	FIFO of size 1 shall be considered as a single buffer Clarification for rule (a): TxConfP shall only be set if <user>If_Transmit returns with success
PDUR244	FIFO of size 1 shall be considered as a single buffer (parameter depth); Clarification (parameter length)
PDUR100	Development error PDUR_E_IF_TX_BUFFER_MISMATCH and PDUR_E_TP_ BUFFER_SIZE_LIMIT removed, “idle” removed at PDUR_E_TP_TX_REQ_REJECTED
PDUR225	Plausibility check for TpChunkSize added
PDUR248	Correction (SduLength required for gateway operation)
PDUR132	SchM_PduR.h, MemMap.h added to include file structure
PDUR142	Clarification: Transmit confirmation is configurable for unbuffered I-PDUs
PDUR194	Clarification: this function must not be called in the context of CanIf_Transmit.
PDUR196	Clarification: this function must not be called in the context of FrIf_Transmit.
PDUR198	Clarification: this function must not be called in the context of LinIf_Transmit.
PDUR102	Correction: Det_ReportError parameter InstanceId added

PDUR425	Reference to ZERO_COST_OPERATION removed
PDUR352	Redundant requirement with no additional functionality was Removed.
PDUR374	Redundant requirement with no additional functionality was Removed.
PDUR396	Redundant requirement with no additional functionality was Removed.
PDUR425 PDUR427	Container names changed to VARIANT-PRE-COMPILE VARIANT-POST-BUILD
PDUR424,PDUR0769, PDUR482,PDUR0767, PDUR0710, PDUR0722, PDUR0724, PDUR0726, PDUR0727, PDUR0732, PDUR0700, PDUR0701, PDUR0733, PDUR0734, PDUR338	<LoTp>_CancelTransmitRequest renamed: <LoTp>_CancelTransmit <LoTp>_CancelReceiveRequest renamed: <LoTp>_CancelReceive <LoTp>_ChangeParameterRequest renamed: <LoTp>_ChangeParameter Synchronicity and reeantrancy corrected
PDUR119	PduR_GetVersionInfo can be called before the PduR is initialized
PDUR551	Correction: <DstLoTp_Transmit> must be called
PDUR654	Renamed PduR_RoutingTableIdType as PduR_RoutingPathGroupIdType
PDUR406; PDUR0772, PDUR0773	Removed return value E_PENDING
PDUR332_Conf	Multiplicity of PduRDestPduHandleId changed to 0..1
PDUR339_Conf	Spelling errors corrected
PDUR631, PDUR632	Restricted multicast Tp transmission to single frames.
PDUR504	Clarification of CDD support
PDUR0670, PDUR100, PDUR106	Production errors PDUR_E_PDU_INSTANCE_LOST and PDUR_E_INIT_FAILED changed to development errors.
PDUR437	Clarification
PDUR512, PDUR507	Move return value BUFREQ_E_OVFL from PduR_<LoTp>CopyRxData to PduR_<LoTp>StartOfReception
PDUR518	Removed retry value TP_NORETRY
PDUR334, PDUR338, PDUR341, PDUR615, PDUR617, PDUR406,	Changed service ids to generic service ids

PDUR0769, PDUR482, PDUR0767, PDUR507, PDUR518, PDUR381	
PDUR303, PDUR306, PDUR640, PDUR307, PDUR0665, PDUR0666, PDUR0667, PDUR255, PDUR0670, PDUR0669	Reworked for RfC #50682 (FIFO Handling)
PDUR0746	Reworked for RfC #48071

13.4 Added SWS Items

SWS Item	*Rationale*
PDUR479	New typedefine PduR_CancelReasonType is added.
PDUR480	New typedefine PduR_ParameterValueType is added.
PDUR481	New API PduR_CancelTransmitRequest is added.
PDUR482	New API PduR_ChangeParameterRequest is added.
PDUR483	New Callback PduR_CanTpChangeParameterConfirmation is added.
PDUR484	New Callback PduR_FrTpChangeParameterConfirmation is added.
PDUR485	New Callback PduR_LinTpChangeParameterConfirmation is added.
PDUR486	New requirement is added to Minimum Routing section.
PDUR403	UML model linking of PduR_LinTpTxConfirmation, Result values are updated.
PDUR397	UML model linking of PduR_LinTpRxIndication, Result values are updated
PDUR381	UML model linking of PduR_FrTpTxConfirmation, Result values are updated.
PDUR375	UML model linking of PduR_FrTpRxIndication, Result values are updated.
PDUR359	UML model linking of PduR_CanTpTxConfirmation, Result values are updated.
PDUR353	UML model linking of PduR_CanTpRxIndication, Result values are updated.
PDUR0778	Rework of Published Information
PDUR0779	Development error for wrong retry parameters in case of a multicast Tp transmission.
PDUR0780, PDUR0781,	Introduced generic service ids

PDUR0782	
PDUR346_Conf	New configuration parameter PduRMaxPduLength
PDUR0783, PDUR0784	for RfC #48071
PDUR0785, PDUR0786, PDUR0787, PDUR0788	RfC #50682 (FIFO Handling)

14 Changes during SWS Improvements by Technical Office

14.1 Deleted SWS Items

SWS Item	*Rationale*
PDUR169	Requirement on the PDU Router module's environment.
PDUR171	No requirement on the module.
PDUR239	No requirement but information
PDUR251	No requirement on the module.
PDUR165	Requirement on Zero Cost Operation
PDUR159	Superfluous, covered by PDUR226
PDUR300	Already covered by PDUR299
PDUR304	No need to require multiple buffers in multicast
PDUR163	Already covered by PDUR161
PDUR297	Meaningless requirement
PDUR298	Not a requirement, each API will specify synch/asynch
PDUR250	Not needed since it is base requirements of the configuratrion
PDUR285	Removed requirement tag since already covered by the configuration requirements
PDUR201	Already covered by PDUR166 and PDUR168
PDUR407	Configuration parameter not needed
PDUR554	No need for configuration parameter that is doing linker job
PDUR590, PDUR593, PDUR594, PDUR595, PDUR596, PDUR591, PDUR597, PDUR598, PDUR599, PDUR600, PDUR592, PDUR601, PDUR602, PDUR603, PDUR604, PDUR605	These requirements are affecting the Dbg interface. Since the APIs are now generic specific Dbg interfaces are not needed.
PDUR413, PDUR237, PDUR238, PDUR414, PDUR415, PDUR416, PDUR417, PDUR418, PDUR419, PDUR420, PDUR421, PDUR422	These requirements are affecting the IpduM interface. Since the APIs are now generic specific IpduM interfaces are not needed.

PDUR408, PDUR409, PDUR410, PDUR411, PDUR206, PDUR412	These requirements are affecting the DCM interface. Since the APIs are now generic specific DCM interfaces are not needed.
PDUR448	Already covered by PDUR166 and PDUR168
PDUR449	No need to reference other requirements
PDUR364	No need for configuration parameter that is doing linker job
PDUR608	Already covered by PDUR608
PDUR450	Already covered by PDUR166 and PDUR168
PDUR451	No need to reference other requirements
PDUR368	TxConfirmation is a mandatory function
PDUR452	Already covered by PDUR166 and PDUR168
PDUR453	Already covered by the gateway section
PDUR371	No need for configuration parameter that is doing linker job
PDUR513	Repetition of PDUR512
PDUR514	Already covered by PDUR166 and PDUR168
PDUR515	Already covered by the gateway section
PDUR517	No need for configuration parameter that is doing linker job
PDUR456	Already covered by PDUR166 and PDUR168
PDUR457	Already covered by the gateway section
PDUR609	Already covered by the I-PDU reception section
PDUR377	No need for configuration parameter that is doing linker job
PDUR508	Already covered by PDUR166 and PDUR168
PDUR509	Already covered by the gateway section
PDUR511	No need for configuration parameter that is doing linker job
PDUR519	Already covered by PDUR166 and PDUR168
PDUR520	Already covered by the gateway section
PDUR548	Replaced by PduR_<bus>TpGetAvailableTxBuffer
PDUR523	No need for configuration parameter that is doing linker job
PDUR460	Already covered by PDUR166 and PDUR168
PDUR461	Already covered by the gateway section
PDUR462	Already covered by PDUR302
PDUR383	No need for configuration parameter that is doing linker job
PDUR105	No requirement
PDUR240	PduR_ReturnType not used
PDUR284	PduR_StateType not used

14.2 Replaced SWS Items

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

PDUR197	PDUR463, PDUR464	Made requirement atomic
PDUR198	PDUR465, PDUR466	Made requirement atomic
PDUR199	PDUR452, PDUR453	Made requirement atomic
PDUR200	PDUR467, PDUR468	Made requirement atomic
PDUR202	PDUR409, PDUR410, PDUR411	Made requirement atomic
PDUR203	PDUR328, PDUR329, PDUR330	Made requirement atomic
PDUR208	PDUR434, PDUR435	Made requirement atomic
PDUR209	PDUR430, PDUR431	Made requirement atomic
PDUR210	PDUR299, PDUR300	Made requirement atomic
PDUR211	PDUR303, PDUR304, PDUR305, PDUR306, PDUR307	Made requirement atomic
PDUR254	PDUR310, PDUR311	Made requirement atomic
PDUR257	PDUR312, PDUR313	Made requirement atomic
PDUR260	PDUR308, PDUR309	Made requirement atomic

14.3 Changed SWS Items

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

SWS Item	***Rationale***
PDUR216	Changed to be independent of bus
PDUR132	Changed to be independent of bus
PDUR161	The PduR shall convert I-PDU IDs not keep a list of I-PDU IDs

14.4 Added SWS Items

SWS Item	***Rationale***
PDUR333	UML model linking of imported types
PDUR334	UML model linking of PduR_Init
PDUR338	UML model linking of PduR_GetVersionInfo
PDUR339	Requirement on PduR_GetVersionInfo

PDUR340	Requirement on PduR_GetVersionInfo
PDUR341	UML model linking of PduR_GetConfigurationId
PDUR342	Requirement on PduR_GetConfigurationId
PDUR343	UML model linking of PduR_CanIfRxIndication
PDUR344	Requirement on PduR_CanIfRxIndication
PDUR345	Requirement on PduR_CanIfRxIndication
PDUR346	UML model linking of PduR_CanIfTxConfirmation
PDUR347	Requirement on PduR_CanIfTxConfirmation
PDUR348	Requirement on PduR_CanIfTxConfirmation
PDUR349	Requirement on PduR_CanIfTxConfirmation
PDUR350	UML model linking of PduR_CanTpProvideRxBuffer
PDUR351	Requirement on PduR_CanTpProvideRxBuffer
PDUR352	Requirement on PduR_CanTpProvideRxBuffer
PDUR354	Requirement on PduR_CanTpRxIndication
PDUR355	Requirement on PduR_CanTpRxIndication
PDUR356	UML model linking of PduR_CanTpProvideTxBuffer
PDUR357	Requirement on PduR_CanTpProvideTxBuffer
PDUR358	Requirement on PduR_CanTpProvideTxBuffer
PDUR360	Requirement on PduR_CanTpTxConfirmation
PDUR361	Requirement on PduR_CanTpTxConfirmation
PDUR362	UML model linking of PduR_FrIfRxIndication
PDUR363	Requirement on PduR_FrIfRxIndication
PDUR364	Requirement on PduR_FrIfRxIndication
PDUR365	UML model linking of PduR_FrIfTxConfirmation
PDUR366	Requirement on PduR_FrIfTxConfirmation
PDUR367	Requirement on PduR_FrIfTxConfirmation
PDUR368	Requirement on PduR_FrIfTxConfirmation
PDUR369	UML model linking of PduR_FrIfTriggerTransmit
PDUR370	Requirement on PduR_FrIfTriggerTransmit
PDUR371	Requirement on PduR_FrIfTriggerTransmit
PDUR372	UML model linking of PduR_FrTpProvideRxBuffer
PDUR373	Requirement on PduR_FrTpProvideRxBuffer
PDUR374	Requirement on PduR_FrTpProvideRxBuffer
PDUR376	Requirement on PduR_FrTpRxIndication
PDUR377	Requirement on PduR_FrTpRxIndication
PDUR378	UML model linking of PduR_FrTpProvideTxBuffer

PDUR379	Requirement on PduR_FrTpProvideTxBuffer
PDUR380	Requirement on PduR_FrTpProvideTxBuffer
PDUR382	Requirement on PduR_FrTpTxConfirmation
PDUR383	Requirement on PduR_FrTpTxConfirmation
PDUR384	UML model linking of PduR_LinIfRxIndication
PDUR385	Requirement on PduR_LinIfRxIndication
PDUR386	Requirement on PduR_LinIfRxIndication
PDUR387	UML model linking of PduR_LinIfTxConfirmation
PDUR388	Requirement on PduR_LinIfTxConfirmation
PDUR389	Requirement on PduR_LinIfTxConfirmation
PDUR390	Requirement on PduR_LinIfTxConfirmation
PDUR391	UML model linking of PduR_LinIfTriggerTransmit
PDUR392	Requirement on PduR_LinIfTriggerTransmit
PDUR393	Requirement on PduR_LinIfTriggerTransmit
PDUR394	UML model linking of PduR_LinTpProvideRxBuffer
PDUR395	Requirement on PduR_LinTpProvideRxBuffer
PDUR396	Requirement on PduR_LinTpProvideRxBuffer
PDUR398	Requirement on PduR_LinTpRxIndication
PDUR399	Requirement on PduR_LinTpRxIndication
PDUR400	Requirement on PduR_LinTpProvideTxBuffer
PDUR401	UML model linking of PduR_LinTpProvideTxBuffer
PDUR402	UML model linking of PduR_LinTpProvideTxBuffer
PDUR404	Requirement on PduR_LinTpTxConfirmation
PDUR405	Requirement on PduR_LinTpTxConfirmation
PDUR406	UML model linking of PduR_ComTransmit
PDUR407	Requirement on PduR_ComTransmit
PDUR408	UML model linking of PduR_DcmTransmit
PDUR412	Requirement on PduR_DcmTransmit
PDUR413	UML model linking of PduR_IpduMTransmit
PDUR414	Requirement on PduR_IpduMTransmit
PDUR415	UML model linking of PduR_IpduMTxConfirmation
PDUR416	Requirement on PduR_IpduMTxConfirmation
PDUR417	Requirement on PduR_IpduMTxConfirmation
PDUR418	Requirement on PduR_IpduMTxConfirmation
PDUR419	UML model linking of PduR_IpduMRxIndication
PDUR420	Requirement on PduR_IpduMRxIndication

PDUR421	Requirement on PduR_IpduMRxIndication
PDUR422	Requirement on PduR_IpduMRxIndication
PDUR423	UML model linking of mandatory interfaces
PDUR424	UML model linking of optional interfaces
PDUR425	Every variant gets a requirement ID
PDUR426	Every variant gets a requirement ID
PDUR427	Every variant gets a requirement ID
PDUR478	An Integrator's remark added for the implementation of critical sections
PDUR487	Added requirement for Debugging concept
PDUR488	Added requirement for Debugging concept
PDUR489	Added requirement for Debugging concept
PDUR490	Added requirement for Debugging concept
PDUR651	Added requirement because of duplicate req number (PDUR507)
PDUR652	Added requirement because of duplicate req number (PDUR507)

15 Not applicable requirements

[PDUR0777] ⌈These requirements are not applicable to this specification.⌋ (BSW170, BSW00398, BSW00375, BSW00416, BSW00437, BSW168, BSW00423, BSW00424, BSW00425, BSW00428, BSW00432, BSW00433, BSW00450, BSW00336, BSW00422, BSW00417, BSW00409, BSW00386, BSW161, BSW162, BSW005, BSW164, BSW00325, BSW00326, BSW00343, BSW160, BSW00413, BSW00347, BSW00373, BSW00335, BSW00447, BSW00348, BSW00353, BSW00361, BSW00439, BSW00449, BSW00357, BSW00376, BSW00443, BSW00444, BSW00445, BSW00446, BSW172, BSW00341, BSW00334, BSW06055, BSW06056, BSW06061, BSW06098, BSW06099, BSW06077, BSW06064, BSW06089)