MICROSAR PDU Router

Technical Reference

Version 3.10.02

Authors	Hartmut Hörner, Hannes Haas, Erich Schondelmaier, Gunnar Meiss
Status	Released

Document Information

History

Author	Date	Version	Remarks
Hartmut Hörner	2006-02-01	0.1	Initial version
Hartmut Hörner	2006-03-13	0.2	Pre-compile variants added in chapter 3.6 Configuration Phases
Hartmut Hörner	2006-03-17	0.3	Added chapter with required API functions
Hannes Haas	2006-06-14	2.0	Adapted to AUTOSAR version 2.0
Hannes Haas	2006-07-30	2.1	Added LIN IF support
Hannes Haas	2006-11-22	2.2	Added CAN TP support Added Post-Build support
Hannes Haas	2007-04-26	2.3	New Template Simplified post-build configuration procedure Revision of all chapters
Hannes Haas	2007-11-20	2.4	Removed non-selectable post-build configuration Added Interface Gateway with Callouts
Hannes Haas	2008-01-08	2.5	Added TP Gateway and memory allocation Added Glossary Adapted to AUTOSAR 3 Added include structure Added J1939TP support
Hannes Haas	2008-03-26	3.0	Corrected return values of callout functions ESCAN00024680: described PDU handle concept Updated to SWS version 2.2.1 TP Gateway Ring-Buffer support ESCAN00025586: changed file and GENy module name
Hannes Haas	2008-05-16	3.1	Minor improvements in all chapters GENy GUI update Added generator version check
Hannes Haas	2008-06-26	3.2	Minor improvements in all chapters Adaptations to new GUI
Hannes Haas	2008-09-25	3.3	ESCAN00028028: Memory mapping limitations Object Explorer Added CRC check for configuration data ESCAN00029973: Adapted GUI according to AUTOSAR short names
Hannes Haas	2008-11-19	3.4	Intra ECU Communication
Erich Schondelmaier	2008-03-26	3.5	Added IpduM support

Hannes Haas	2009-08-03	3.6	TP Layer Routing using AUTOSAR 4.0 API
Gunnar Meiss	2009-08-30	3.6	Timeout in Intra ECU Communication
	2009-08-12	3.6	Added LIN TP support Added MOST IF support Added MOST TP support
Gunnar Meiss	2009-10-12	3.7	Added SoAd support Added Dobt support
Erich Schondelmaier	2010-01-08	3.8	Added Nm support Added Service Ids
Erich Schondelmaier	2010-03-30	3.9	Updated 6.2.4 TP Buffers
Erich Schondelmaier	2010-04-29		Renamed J1939 to J1939Tp Added 3.14 AUTOSAR 4.0 CAN TP API Added 3.13 AUTOSAR 3.0 TP API with ISO timing optimization
Gunnar Meiss	2010-08-23		Updated to new Template Added PduR_EnableRouting and PduR_DisableRouting Added PduR_<UpTp>CancelReceiveRequest Added PduR_<UpTp>CancelTransmitRequest Added PduR_<UpTp>ChangeParameterRequest for CanTp and FrTp Added PduR_<UpTp>ReadParameterRequest Added chapter 6.5 Configuration in EcuC Data Base
Erich Schondelmaier	2010-09-23		Added new BSWMD Parameter Change Functional Description of PduR_Init Updated AUTOSAR 3.0 TP API with ISO timing optimization
Gunnar Meiss	2010-10-19	3.10.00	Added Rx for CanNm and FrNm
Erich Schondelmaier	2010-12-20		Added description for Balance Routing time of Physical TP Routings Added chapter Tp error handling Updated API descriptions
Gunnar Meiss	2011-02-18		Added Cdd Added Dynamic DLC Routing
Gunnar Meiss	2012-08-10	3.10.01	AR3-2457: Dynamic DLC with TriggerTransmit based Communication Interfaces Added partial deviation of PDUR350 (Can), PDUR372 (Fr), PDUR394 (LIN)
Erich Schondelmaier	2012-10-31	3.10.02	Added decription of ESCAN00062517

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_PDU_Router.pdf	2.2.1
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	2.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	2.2.1
[4]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	1.3.0
[5]	AUTOSAR	AUTOSAR_SRS_Gateway.pdf	2.0.3
[6]	AUTOSAR	AUTOSAR_SWS_PDU_Router.doc	3.8.1 (draft)
[7]	Vector	Technical Reference Post-Build Tool Chain	

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
2.00.00	> Support for AUTOSAR release 2.0
2.01.00	> LIN Interface support added > CAN TP support added
2.02.00	> Post-build configuration
2.03.00	> Support for AUTOSAR release 2.1 > FlexRay Interface and TP support
2.04.00	> Interface layer gateway
2.05.00	> Support for AUTOSAR release 3 (partially) > TP layer gateway > J1939Tp support > API forwarding between COM and TP layer > AUTOSAR memory and compiler abstraction
3.00.00	> API change for COM <-> CanTp API forwarding > AUTOSAR release 3 (PDU Router SWS Version 2.2.1) > TP layer routing supporting ring-buffer
3.01.00	> Minor improvements (see source code history)
3.02.00	> Support for TMS320 > Generator version check
3.03.00	> Interface for application access to interface and TP layer PDUs (optional AUTOSAR extension)
3.04.00	> CRC Check of configuration data > Application access to PDU Router services using the modules ApplIf and ApplTp
3.05.00	> Intra ECU communication for TP PDUs
3.06.00	> IPDUM support added
3.07.00	> MOST Interface support added > MOST TP support added
3.08.00	> Multiple Configuration
3.09.00	> LIN TP support added > Optional CanTp API as specified by AUTOSAR 4.0 > MOST TP support added
3.10.00	> SoAd TP support added
3.11.00	> API forwarding between COM and CanNm, FrNm layer

Component Version	New Features
	> CddDobt support added
3.12.00	> Minor Changes
3.13.00	> Minor Changes
3.14.00	> TP Receive Cancellation for CanTp, LinTp, FrTp > TP Transmit Cancellation for CanTp, LinTp, FrTp > PduRoutingPathGoups > Support for CanNm, FrNm and Nm_DirOsek
3.15.00	> Support Rx for CanNm and FrNm > Dynamic DLC Support > Added Generic Cdd Interface
3.15.09	> AR3-2457: Dynamic DLC with TriggerTransmit based Communication Interfaces

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module PDUR as specified in [1].

Supported AUTOSAR Release*:	3	
Supported Configuration Variants:	link-time, post-build	
Vendor ID:	PDUR_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	PDUR_MODULE_ID	51 decimal (according to ref. [4])

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The main task of the PDU Router module is to abstract from the type of bus access (Interface layer and TP layer) and from the bus type itself. Besides this, the PDU Router module provides interface layer (low-level message) gateway and TP (high-level) gateway functionality. The general requirements for the gateway functionality are described in [5].

Since the PDU Router module has to route Rx and Tx PDUs to and from the upper- and lower- layers and any software component uses its own handle space, multiple routing tables are required. The PDU Router uses the input handle as an index to the related routing table.

Info
The PDU Router does not forward I-PDUs to the NM and TP layer. According to AUTOSAR [1] the interface layer (e.g. CAN IF) has to forward these I-PDUs to the NM or TP module directly.

2.1 Architecture Overview

The following figure shows where the PDUR is located in the AUTOSAR architecture.

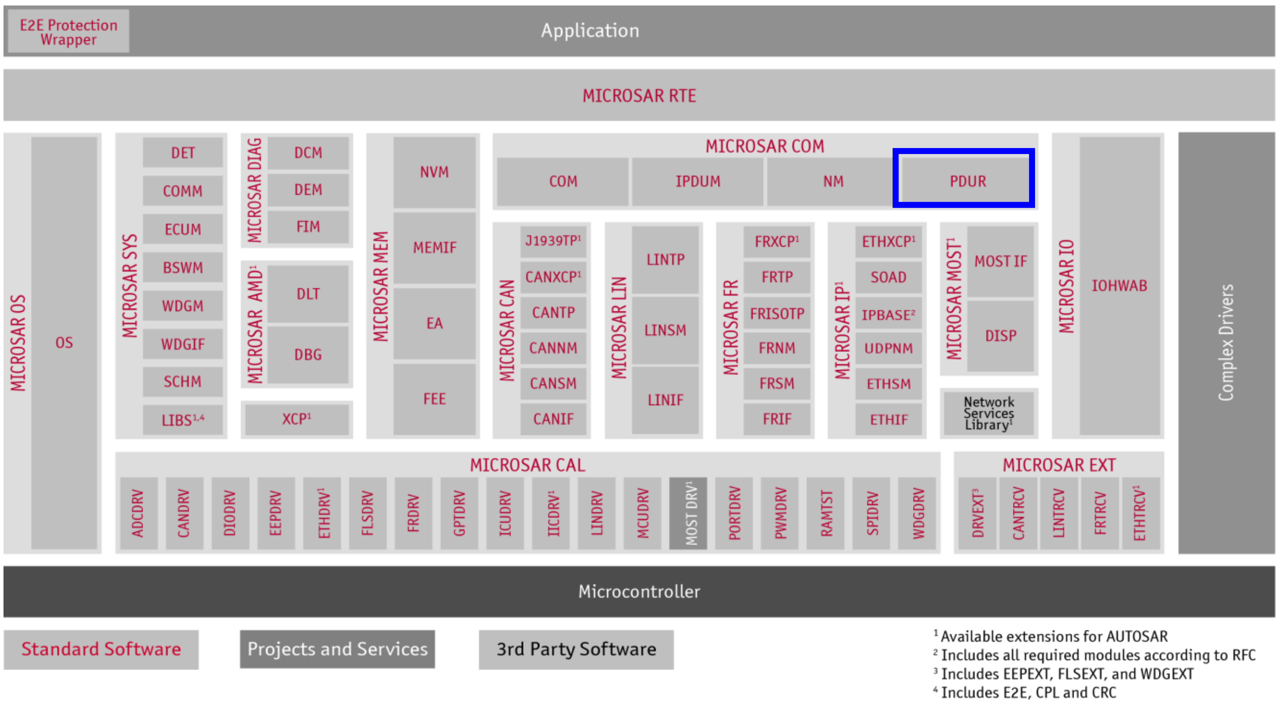


Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the PDUR. These interfaces are described in chapter 5.

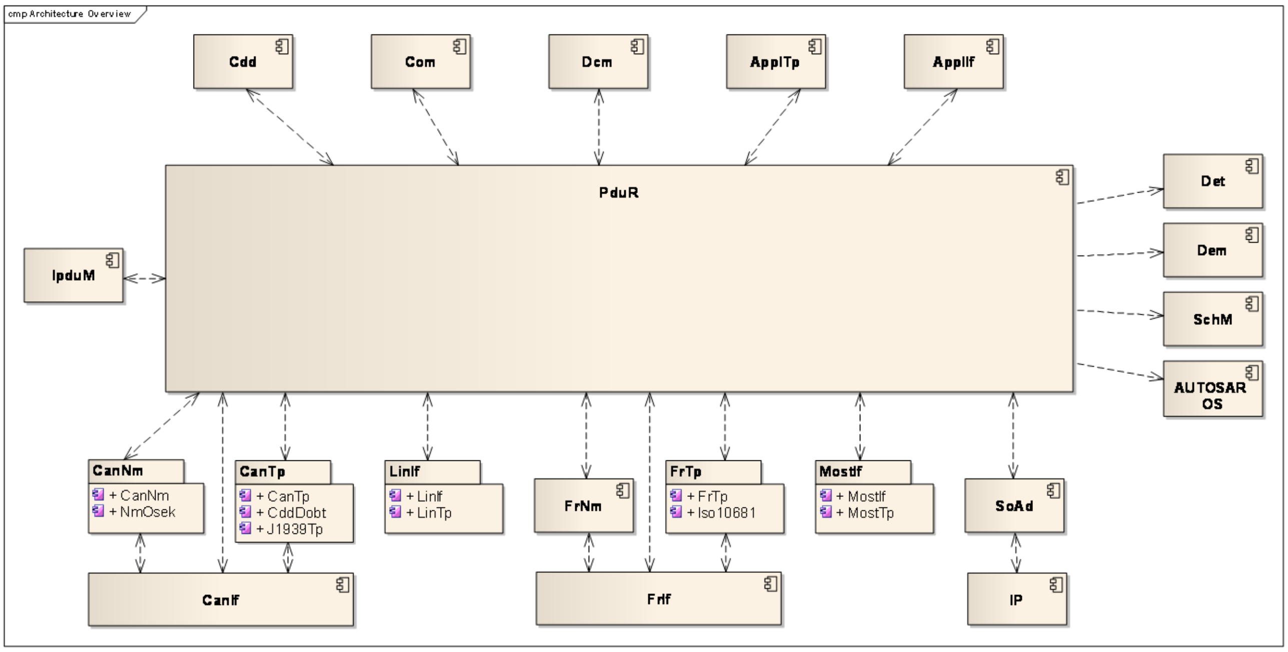


Figure 2-2 Interfaces to adjacent modules of the PDUR

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features also see chapter 0.

The following features described in [1] are supported:

Supported Feature
Link time configuration
Post-build time configuration
I-PDU transmission with and without confirmation and I-PDU reception
ECU Configuration file import and export
1:1 routing between upper- and lower- layer BSW modules
Interface Gateway Routing > 1:1 routing > 1:N routing > Support of “trigger-transmit” and “direct” data provision > FiFo queued routing
Transport Protocol Routing > 1:1 routing of physical requests > 1:N routing of functional (single frame) requests > Forwarding of functional (single frame) requests to DCM and routing of the same Rx I-PDU > Ring-buffer concept
PduR_GetVersionInfo
PduR_GetConfigurationId
PduR_CancelTransmitRequest
PduR_DcmChangeParameterRequest
Dcm_ChangeParameterConfirmation

Table 3-1 Supported SWS features

The following features are additionally supported:

Supported Feature
IF Routings to MOST, CanNm, FrNm, NmOsek
TP Routings to Com, MOST, CddDobt, ISO10681, SoAd, J1939TP
Application IF and TP I-PDU Access including the prototypes of required callback functions.

Supported Feature
TP Intra ECU Communication between DCM and ApplTp (PduR_MainFunction)
Callout functions for IF Gateway Routing Paths
PduR_InitMemory() API
AUTOSAR TP layer CanTp to CanTp routing according to AUTOSAR 4.0 (draft) specification
PduR_<UpperLayer>CancelReceiveRequest
Partial Networking
Complex Device Driver Upper Layer Interfaces

Table 3-2 Additionally supported features

The following features described in [1] are not supported:

Not Supported Feature
TP layer fan-out
Zero cost operation
Minimum Routing (Reduced state)

Table 3-3 Not supported SWS features

3.2 Initialization

Before the Pdu Router can be used it has to be initialized by PduR_Init() which performs the basic initialization. Initialization, starting and stopping of the layer and its I-PDU groups is normally driven by the Communication Manager. If this software component is not available a similar component has to be provided by the integrator.

3.3 States

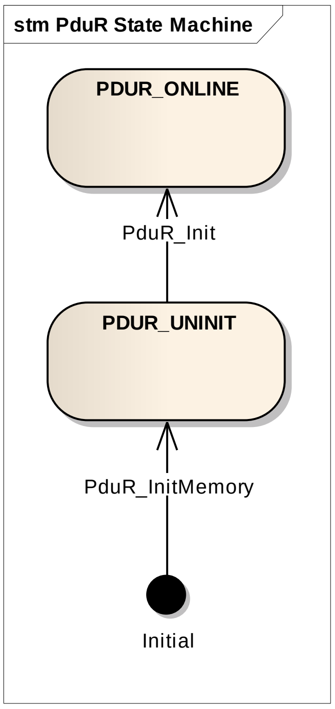


Figure 3-1 PduR State Machine

The PduR is initially not activated. Basic RAM arrays are initialized with the call of PduR_InitMemory or with the startup code of your ECU. If PduR_Initi is called with valid parameters, the PduR is in the state PDUR_ONLINE and the communication can start.

3.4 Main Functions

The PduR provides one function that has to be called cyclically by the Basic Software Scheduler or a similar component.

Main Functions	Description
PduR_MainFunction()	The main function is only used, if the TP intra ECU communication is used. Depending on the size of the I-PDU to be routed it is therefore important to run the main function in a low priority task with a sufficient low call frequency. See chapter 3.16 TP Intra ECU Communication and 5.1.5 PduR_MainFunction

Table 3-4 Main functions that have to be called cyclically

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `PDUR_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported PDUR ID is 51.

The reported service IDs identify the services which are described in 5.1. Due to this, that the API is partially generated, the servide IDs are assigned dynamically at generation time. Please use the symbolic names generated for the services.

The errors reported to DET are described in the following table:

ID		Description
((PduR_Core_ErrorType)0x00u)	PDUR_E_CONFIG_PTR_INVALID	This error code is only used in conjunction with post-build configuration: An invalid post-build configuration data pointer has been handled to PduR_Init(). The error is either caused by a failed consistency check or because a null pointer was handled to PduR_Init().
((PduR_Core_ErrorType)0x01u)	PDUR_E_INVALID_REQUEST	An API service has been used without a call of PduR_Init() or PduR_Init() was called while the PDU Router is in any other state than PDUR_UNINIT. If your system does not use a start-up code to initialize global variables, PduR_InitMemory() can be called before PduR_Init() to avoid this problem.
((PduR_Core_ErrorType)0x02u)	PDUR_E_PDU_ID_INVALID	An invalid PDU identifier was used as parameter for a PDU Router API call.
((PduR_Core_ErrorType)0x03u)	PDUR_E_TP_TX_REQ_REJECTED	TP Gateway was not able to transmit a TP Tx I-PDU due to an error reported by the TP layer. The gateway will abort the routing and not attempt a re-transmission.
((PduR_Core_ErrorType)0x05u)	PDUR_E_DATA_PTR_INVALID	The Data pointer (CanSduPtr, FrSduPtr, LinSduPtr or PduInfoPtr) is a NULL_PTR
((PduR_Core_ErrorType)0x06u)	PDUR_E_TP_BUFFER_SIZE_LIMIT	Length of requested TP buffer is larger than the maximum length of all configured TP buffer. The gateway will

ID		Description
		try to route the I-PDU using the ring-buffer mechanism. This can, however, result in a buffer overrun if the TP layer is not able to adapt the block size of the Rx connection.
((PduR_Core_ErrorType)0x07u)	PDUR_E_UL_BUFFER_UNDERRUN	The provided buffer of the upperlayer module is to small, should be minimum the size of the CanTp FF + minimum 1 byte. It is required by the CanTp
((PduR_Core_ErrorType)0x08)	PDUR_E_ROUTING_TABLE_ID_INVALID	If the routing table is invalid that is given to the PduR_EnableRouting or PduR_DisableRouting functions
((PduR_Core_ErrorType)0xA0u)	PDUR_E_INVALID_CFG_DATA	An internal PDU Router error occurred that was eventually caused by incorrect or manipulated configuration data.
((PduR_Core_ErrorType)0xA1u)	PDUR_E_UNEXPECTED_CALL	The indicated API was called although the current PDU Router configuration and internal state does not expect a call to this API.
((PduR_Core_ErrorType)0xA2u)	PDUR_INCONSISTENT_SIZEOFPDUIDTYPE	The size of PduIdType is consistent

Table 3-5 Errors reported to DET

3.5.2 Production Code Error Reporting

By default, production code related errors are reported to the DEM using the service `Dem_ReportErrorStatus()` as specified in [3], if production error reporting is enabled (i.e. pre-compile parameter `PDUR_PROD_ERROR_DETECT==STD_ON`).

If another module is used for production code error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Dem_ReportErrorStatus()`.

The errors reported to DEM are described in the following table:

Error Code	Description
PDUR_E_INIT_FAILED	Post-build configurations only: Invalid post-build configuration data pointer detected. Either the consistency check failed during PduR_Init() or a null pointer has been used as configuration structure reference.

PDUR_E_PDU_INSTANCE_LOST	This error is reported in case the transmission request of a FiFo queued I-PDU failed or in case the queue was flushed due to an overrun. These events cause the loss of one or several I-PDU instances which are routed using a FiFo queue. The error is not reported for I-PDUs that are routed without a FiFo algorithm.

Table 3-6 Errors reported to DEM

Info
The error code symbols have to be defined by the DEM module.

3.6 Configuration Phases

3.6.1 Link-time

This step has to be executed independently of the PDU Router configuration variant.

Link-time parameters are used to configure the PDU Router according to the vehicle specific communication properties with respect to the number of used I-PDUs and the routing properties of each I-PDU.

The result of link-time configuration is a set of routing tables generated to PduR_PBcfg.c. Link time specific parameters that cannot be changed at post-build time are generated to PduR_Lcfg.c.

At link-time the following properties can be changed:

- Number of I-PDUs that are used by each layer.
- The destination I-PDU handle and the destination BSW module (destination port) of each routing path.
- Number of interface gateway routing paths.
- Number and properties of TP gateway routing paths and TP buffer elements.
- Number and properties of application PDUs
- Number of concurrent intra ECU communication connections (see chapter 3.16 TP Intra ECU Communication for details)
- Maximum number of Tx <Up> (DCM, COM, ApplTp) connections via CanTp
- Maximum number of Rx CanTp connections used by upper layers

3.6.2 Link-time for post-build capable configurations

If the PDU Router module supports post-build configuration, further link time settings are required. These settings are required to allow the PduR configuration to be adapted at post-build time:

> Configuration ID: According to AUTOSAR [1] each configuration can be labeled with a unique 32bit identifier that can be retrieved by the PduR_GetConfigurationId() API. For the initial configuration that is set up at link-time, this ID can be configured in the GENy PDU Router GUI.

> Maximum Tx Buffer Number: This GENy attribute specifies maximum number of FiFo queues of interface layer routing paths that shall be available at post-build time. Allow unused queue elements in case adding FiFo queued routing paths shall remain an option at post-build time.

> Maximum TP Buffer Number: To allocate RAM resources for the buffer state machines the maximum number of TP buffers has to be configured at link time.

> Memory Size [bytes]: This attribute specifies the RAM that will be made available to the PDU Router module. In order to allow adding further routing paths at post-build time or to increase the queue depth, it is necessary to reserve additional RAM resources at link-time. Routing relations that require additional RAM resources at post-built time are e.g. single buffer or FiFo queued routing paths. The currently required memory size can be calculated automatically using the button “Compute Memory Size” or can be determined according to chapter 3.6.

> COM Generate: This is a COM attribute. Disable the PDU specific checkbox “Generate” for all Tx I-PDUs that are not required by COM but may be part of a routing path at post-build time. It will not be possible to remove the Generate attribute at post-build time. Tx PDUs that are part of a PDU Router interface layer routing path must not be handled by COM (COM Generate is disabled).

3.6.3 Post-build

Configuring the PDU Router module at post-build time is only possible if the PDU Router module supports the configuration variant “Post-build”.

The configuration at post-build time is similar to the link-time configuration. Interface-, TP layer routing paths and TP buffer elements can be configured as long as enough (RAM) resources are available.

The module start address must be used to specify the start address of the PDU Router configuration data on the target ECU. The address is required for the post-build configuration process. For further details please see the post-build configuration documentation [7].

At post-build time the following properties can be changed:

> Number of I-PDUs that are used by each layer.

> Number of interface layer routing paths and their properties (e.g. queue depth or SDU length). The maximum number of routing paths is limited by the link-time parameter PDUR_MEMROY_SIZE (GENy: “Memory Size [byte]”).

> Number of TP layer routing paths and their properties (e.g. Chunk Size). The maximum number of routing paths is limited by the link-time parameter PDUR_MEMROY_SIZE (GENy: “Memory Size [byte]”).

> Number of TP buffer elements and the buffer size. The maximum number of buffer elements is limited by the link-time parameter PDUR_MAX_TX_BUFFER_NUMBER (GENy: “Maximum TP Buffer Count”). The sizes of the buffer elements is limited by PDUR_MEMROY_SIZE (GENy: “Memory Size [byte]”).

> The destination I-PDU handle and the destination BSW module (destination port) of each routing path.

> Number and properties of application PDUs

> Post-build configuration ID.

> The start address of the module configuration in ROM can be changed at post-build time. During PDU Router initialization a pointer to the configuration structure must be provided.

Caution
Tx I-PDUs that are part of an interface layer routing path are no longer handled by COM. COM Generate for Rx I-PDUs can optionally be removed in case the gateway ECU application does not require their data.

A Tx I-PDU must not be added to a routing path at post-build time in case the I-PDU was handed by COM at link time. Adding such an I-PDU to a routing path would cause changes to the COM signal handle space as COM would no longer consider the signals of Tx I-PDUs which are part of a routing path.

In case of adding an Rx I-PDU to a routing path at post-build time, the COM Generate for the particular I-PDU must remain as it was in during link-time.

3.7 Memory allocation

As adding of routing paths at post-build time is not possible using the link-time configuration variant, this section is not required if the PDU Router module does support link-time configuration only. In this case the PDU Router generator is able to compute the required memory size automatically at generation time.

In case of a PDU Router module supporting post-build configuration, routing paths and TP buffers can be added at post-build time. Depending on the added configuration element, this causes an increased RAM usage of the PDU Router module. To support this, additional RAM has to be allocated at link time, using the “Memory Size [byte]” attribute of the PDU Router configuration dialogue in GENy.

Using the button “Compute Memory Size” the currently required memory size is computed and written to the “Memory Size [byte]” attribute. The following table allows estimating PDU Router buffer requirements in case of post-build time modification. The memory that shall be reserved for post-build modification has to be added to the memory required at link-time.

By deleting or re-configuration at post-build time, it is also possible to free memory. Memory that has been freed at post-build time is usable for other PDU Router configuration elements.

Feature	Buffer requirement [byte]
Interface layer routing path with direct data provision (no queuing)	0
Interface layer routing path with direct data provision (FiFo queued)	Size of routed I-PDU multiplied by the queue depth
Interface layer routing path with trigger-transmit data provision (no queuing)	Size of routed I-PDU
Interface layer routing path with trigger-transmit data provision (FiFo queued)	Size of routed I-PDU multiplied by the queue depth
TP buffer element	Configured size of the TP buffer element
TP layer routing path	1
Application Access for I-PDUs	0

Table 3-7 Memory usage (RAM) of different PDU Router features (relevant for computing the memory size required by PDU Router)

3.8 Interface Layer Gateway

The interface layer gateway allows low-level routing of I-PDUs provided by the interface layer. If loss of data is critical, routing paths can be configured to use a Tx PDU specific FiFo queue.

3.8.1 Dynamic Dlc Routing

The standard behaviour of the Interface Layer Gateway is to route all the received data defined at build time. If your delivery is tailored to use the PduInfoPtr as parameter as specified in ASR 4 and the dynamic dlc support is configured in the PduR the TX DLC depends on the RX DLC, TX DLC and the intitial build in TX DLC in the configuration.

DataProvision of the Routing Destination	RoutingType	Description
DIRECT e.g. CanIf	No Buffer	The TX DLC is equal to the RX DLC and the RX DLC can be larger than in the initial build in DLC.
DIRECT e.g. CanIf	FIFO Queue	The TX DLC is equal or smaller than the the RX DLC. If the RX DLC is smaller or equal to the initial build in DLC the TX DLC equals to the RX DLC. If the RX DLC is greater than the initial build in DLC the TX DLC equals to the initial build in DLC and the data is shortened.
TRIGGER_TRANSMIT e.g. LinIf, FrIf	Single Buffer	The TX DLC is equal to the initial build in DLC. If the RX DLC is smaller to the initial build in DLC the remaining data is defined by the default value. If the RX DLC is greater than the initial build in DLC the TX DLC equals to the initial build in DLC. If the RX DLC is greater than the initial build in DLC the TX DLC equals to the initial build in DLC and the data is shortened.
TRIGGER_TRANSMIT e.g. LinIf, FrIf	FIFO Queue	The TX DLC is equal to the initial build in DLC. If the RX DLC is smaller to the initial build in DLC the remaining data is defined by the default value. If the RX DLC is greater than the initial build in DLC the TX DLC equals to the initial build in DLC. If the RX DLC is greater than the initial build in DLC the TX DLC equals to the initial build in DLC and the data is shortened.

Table 3-8 Dynamic DLC Interface Routing Overview

Caution
The variability of the RX DLC is limited and shall be changed by different network setups in Car variants and not with each reception of a message.

3.9 Low-Level routing of TP N-PDUs

If the TP segments (N-PDUs) on the source and the destination network are identical, it is possible to route TP I-PDUs using the interface layer gateway (“low-level” routing). If low-level routing is used, the (former) N-PDU is no longer accessible to the TP layer and therefore seen as an I-PDU by the PDU Router module.

The advantage of low-level routing is that it is executed in the context of the interface layer RxIndication and therefore introduces a minimal routing latency.

Low-level routing has, however, several drawbacks which might cause that a high-level TP routing is more adequate:

> TP protocol conversion is not possible as the frame-layout and the flow-control handling must be the same on the source and the destination network.
>
> Each routing path requires a dedicated FiFo buffer. This results in a high RAM consumption. The TP layer gateway shares its buffer elements among the TP routing paths.
>
> No forwarding of routed TP I-PDUs to the local DCM is possible as it may be required for functional requests.
>
> Eventual loss of TP parameters, such as the STmin timing and the block size, due to bursts on the destination bus. Bursts are a result of queued I-PDUs which were transmitted in the TxConfirmation of the previous I-PDU.
>
> A buffer overrun in the FiFo queue causes the queue to be flushed and therefore corrupts the TP communication. The TP layer gateway can avoid buffer overflows if the receiving TP connection supports a dynamic block size adaptation.

3.10 Buffer Types

In total four different buffer types are supported by the PDU Router. At configuration time the only relevant input is the queue depth of the buffer. The queue depth specifies whether a FiFo queue is used for the routing path or if the most recently received data is transmitted. The queue implementation is based on [1].

GENy automatically detects the type of data provision required by the Tx I-PDU (trigger-transmit or direct data provision).

3.10.1 Buffer overwrite routing (queue size < 2)

If the queue depth is set to 1 or 0, the most recent data of each I-PDU is transmitted at any time. If an I-PDU instance is received, the previous I-PDU instance is overwritten; regardless of the former instance being transmitted or not.

3.10.1.1 Timing aspects

The transmission is triggered directly within the Rx indication of the source I-PDU. Depending on the type of bus access the transmission might be delayed or not occur at all (e.g. in case of a not appropriate LIN schedule table).

The PDU Router does not provide a mechanism to implement a rate conversion (e.g. change the cycle time from the source to the destination channel). A rate conversion can be implemented (at extra runtime costs) by signal routing paths using the COM signal gateway.

3.10.1.2 Direct data provision

Direct data provision is chosen for CAN I-PDUs only. The PDU Router does not provide a buffer for such routing paths (AUTOSAR: “no buffer”) as the interface layer copies the data within the transmission request call.

3.10.1.3 Trigger transmit data provision

The interface layer requests the data through a trigger-transmit call from the PDU Router. To provide the routed data asynchronously, PDU Router uses an own buffer containing a single I-PDU instance (AUTOSAR: “single buffer”). The buffer always contains the latest received data. If no reception has taken place yet, the PDU Router will copy the default value of the Tx I-PDU to the interface layer.

3.10.2 FiFo queued routing (queue size >= 2)

FiFo queued routing is used if loss of I-PDU instances is critical. The PDU Router provides a dedicated FiFo queue for each Tx I-PDU – independent of the communication channel.

In case of several queued I-PDUs (with a different I-PDU ID), the order of transmission depends upon the bus access of the driver layer and not on the relative order of I-PDU reception. The FiFo queue therefore only cares for a FiFo based sorting of the instances of a one I-PDU. The queue depth can be configured for each routing path independently.

If the transmission of an I-PDU failed, (negative return value of the interface layer transmit request), PDU Router removes the I-PDUs’ instance from the queue and retries the transmission with the next instance – until the queue is empty or the transmission request is accepted.

In case of a buffer overrun, all queued I-PDU instances of the affected queue are removed and the newly received I-PDU is transmitted.

3.10.2.1 Timing aspects

The PDU Router triggers the transmission of I-PDU instances to be routed as soon as possible. If the queue is empty, a reception will directly cause a transmission request to the interface layer. If the queue is occupied, the I-PDU instance will be added to the queue. Queued I-PDU instances are transmitted within the Tx confirmation of the preceding instance.

This queuing behavior can cause bursts on the destination channel (especially CAN) if several queue instances are queued and if the driver layer does not eliminate them.

A rate conversion or assurance of certain STmin timing is not possible.

3.10.2.2 Example for the FiFo queue handling

The following example assumes four instances (A-D) of two CAN I-PDUs that are transmitted on the same destination bus. The source I-PDUs are received at a recently higher rate than they can be transmitted on the destination bus. Due to this, the I-PDU instance A (ID 0x0B0) is transmitted directly after reception while all other I-PDU instances are queued to the I-PDU specific FiFo queues (instances B and D) or to the I-PDU specific Tx buffer within the CAN IF (instance C).

Within the confirmation function of instance A, the PDU Router will trigger the transmission of instance B, which is written to the CAN IF transmit buffer as the transmission of instance C is also pending, however instance C (0x0A1) has a higher priority than instance B (0x0B0). The CAN IF will therefore trigger the transmission of instance C.

Within the confirmation of instance C, PDU Router will trigger the transmission of instance D. As instance D has still a higher priority than the also buffered instance B, instance D will be transmitted directly by the CAN IF.

As the queue of the I-PDU with CAN ID 0x0A1 is empty, no further transmission of this I-PDU will be triggered in the following confirmation. This allows the transmission of instance B which is still awaiting transmission.

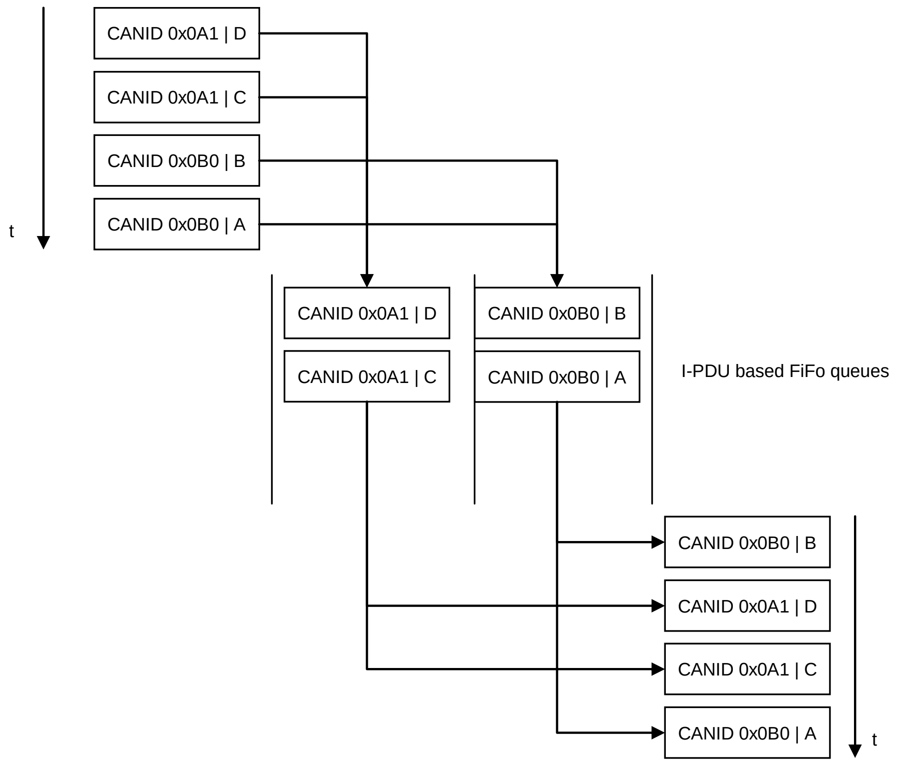


Figure 3-2 Usage of the FiFo buffer (D FiFo and TT-FiFo handling)

3.11 Transport Protocol Gateway

The TP layer gateway allows high-level routing of TP I-PDUs. In order to reduce runtime and memory consumption, the gateway supports the so called on-the-fly routing. Depending on the “Chunk Size” configuration of each routing path, the gateway starts with the transmission on the destination network before the reception has completed on the source network.

Due to the fact that diagnostic communication usually does not take place with all ECUs at the same time, a TP routing path is not associated to a dedicated buffer. In order to reduce the amount of RAM required for queues, the buffers (TP buffer elements) are assigned dynamically to active TP routing paths. If all available buffers of the required size are occupied, no further routing can take place. The TP buffers are freed again after completion of the routing.

A TP routing path has no reference to a TP buffer. Instead, TP buffers have to be configured separately (see chapter 6.2.4 TP Buffers). At runtime the gateway will assign a suitable buffer to an incoming TP routing path (see chapter 6.2.4). If there is no unused TP buffer element available or the available TP buffers are smaller than the chunk size of the routing path, no routing will be executed. The gateway signalizes the state “BUFREQ_E_BUSY” towards the TP module that will, depending on its capabilities, create an appropriate flow-control frame.

The TP buffer is released during initialization of PDU Router module and after the routing has terminated successfully or with an error.

Info
Depending on the number of routing paths that shall be handled by the gateway synchronously, the number of configured TP buffer elements has to be configured.

It might be sensible to provide a small TP buffer (e.g. 7 bytes) to allow single frame routing without occupying a larger and therefore more “expensive” buffer for this task.

3.12 AUTOSAR 3.0 TP API

Using the API described in [4], the TP BS will be determined solely on basis of the TP N-SDU configuration and the currently available TP Buffer size.

If the receiving TP connection can adapt the block size (BS) dynamically (BS > 0), the TP chunk size has a direct affect on the BS of the receiving connection. In this case, the receiving TP connection will reduce its BS according to the buffer that has been made available in order to avoid buffer overflows.

A small BS therefore results in more flow-control (FC) frames. To reduce the number of FC frames the TP chunk size can in turn be increased to allow more data to be received during one TP block.

AUTOSAR does not specify any minimum chunk size that has to be handed to the TP layer by the PDU Router. However, TP module implementation and runtime behavior suggest a minimum sensible chunk size. The following assumptions have been agreed within Vector Informatik and do not necessarily match with third party modules.

The smallest sensible chunk size for I-PDUs that are transferred in segments is the size of one Rx connection first frame (FF) plus one consecutive frame (CF) payload for CAN Rx

TP I-PDUs. This size allows the CAN TP module to transmit a FC frame with a BS of 1 after receiving a FF. When using Rx I-PDUs provided by the AUTOSAR 3.0 FrTp, the chunk size must be at least the size of one CF payload. When using an FrTp according to ISO10681-2, the chunk size can be of any size greater 0.

A smaller chunk size would result in unnecessary buffer requests from the TP layer to the PDU Router module. This is currently not supported by the TP modules. Routing relations between I-PDUs that are transferred as SF only require a TP chunk size of at least one Rx SF payload.

3.13 AUTOSAR 3.0 TP API with ISO timing optimization

If the feature to fulfill ISO timing requirements and constant block size is activated in your delivery PDUR_<LoTp>2PROVIDERXBUFFER_CALLS is generated in PduR_Cfg.h to STD_ON. This is done to route a FF from a CanTp or FrTp to another Tp as fast as possible and to provide a constant TP block size. The PduR returns with the first call of `PduR_<Lo>TpProvideRxBuffer()` always a chunk with the size of the configured chunk size. The pointer to the PduR Tp buffer will be updated depending on the `PduInfoPtr->SduLength` of the first `PduR_<Lo>TpProvideRxBuffer()` call. The wrap around in the PduR buffer is detected dynamically by the PduR at runtime. There are possibly remaining bytes in the buffer due to the last Chunk might not fit into the buffer caused by the first call of `PduR_<Lo>TpProvideRxBuffer()` with 6 Bytes data. The CanTp or FrTp calls for the FF `PduR_<Lo>TpProvideRxBuffer()` twice. In the first call of `PduR_<Lo>TpProvideRxBuffer()` the <LoTp> copies 6 Bytes data and confirms the copy by changing the `PduInfoPtr->SduLength`. The second call of `PduR_<Lo>TpProvideRxBuffer()` will be performed directly after the first call. This is the trigger to call the `<Lo>Tp_Transmit()` function on the destination network. Both `PduR_<Lo>TpProvideRxBuffer()` calls will be performed in the same task context.

The routing of CFs is triggered if the chunk is received completely. Please take the chunk size into account for the calculation of the CanTp timing parameters ($ST_{min,}$ Nas, Nbs, Ncs).

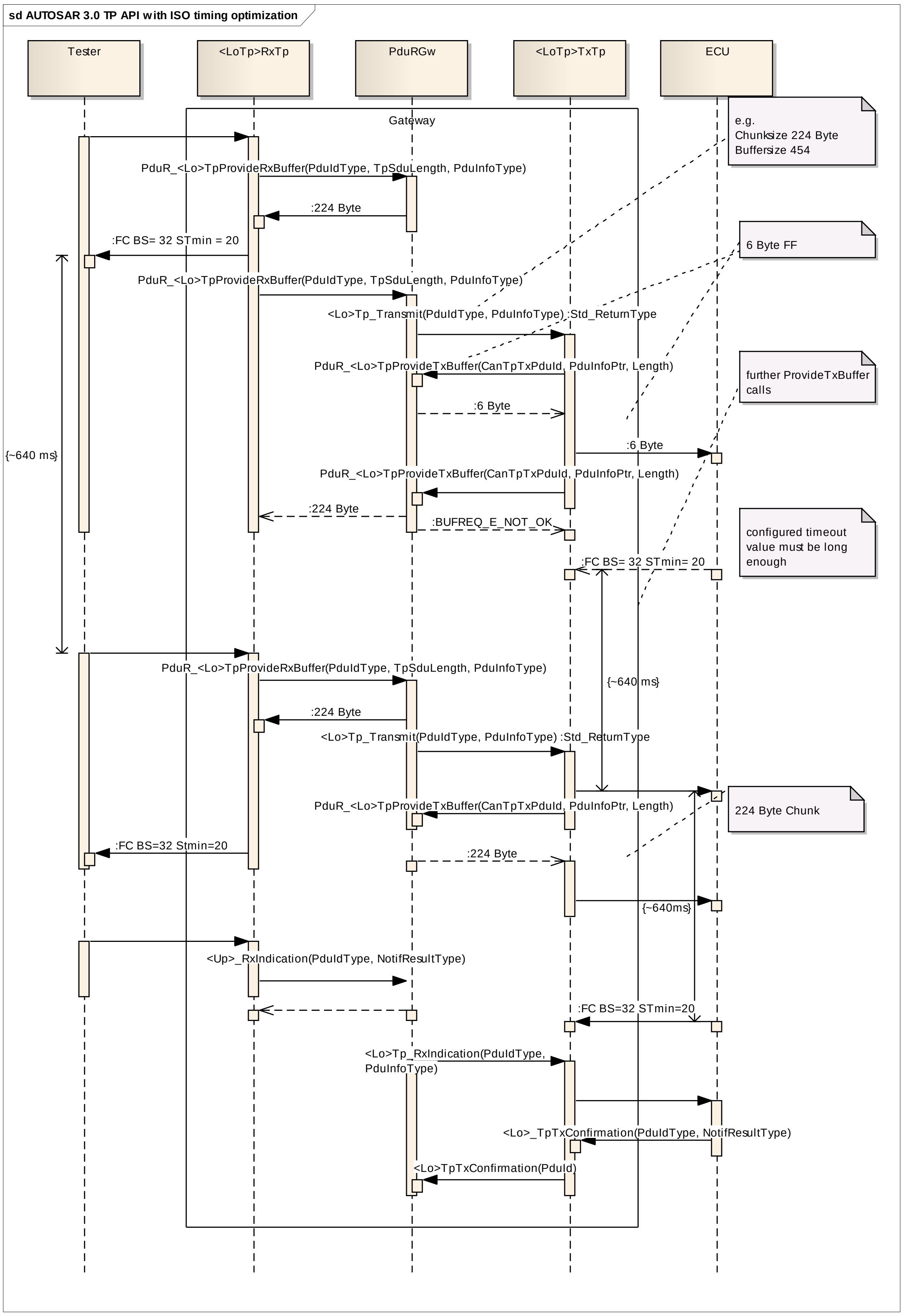

sd AUTOSAR 3.0 TP API with ISO timing optimization
Tester
<LoTp>RxTp
PduRGw
<LoTp>TxTp
ECU
e.g.
Chunksize 224 Byte
Buffersize 454
Gateway
PduR_<Lo>TpProvideRxBuffer(PduIdType, TpSduLength, PduInfoType)
:224 Byte
:FC BS= 32 STmin = 20
6 Byte FF
PduR_<Lo>TpProvideRxBuffer(PduIdType, TpSduLength, PduInfoType)
<Lo>Tp_Transmit(PduIdType, PduInfoType) :Std_ReturnType
PduR_<Lo>TpProvideTxBuffer(CanTpTxPduId, PduInfoPtr, Length)
further ProvideTxBuffer calls
:6 Byte
{~640 ms}
:6 Byte
PduR_<Lo>TpProvideTxBuffer(CanTpTxPduId, PduInfoPtr, Length)
:224 Byte
:BUFREQ_E_NOT_OK
configured timeout value must be long enough
:FC BS= 32 STmin= 20
PduR_<Lo>TpProvideRxBuffer(PduIdType, TpSduLength, PduInfoType)
{~640 ms}
:224 Byte
<Lo>Tp_Transmit(PduIdType, PduInfoType) :Std_ReturnType
PduR_<Lo>TpProvideTxBuffer(CanTpTxPduId, PduInfoPtr, Length)
224 Byte Chunk
:FC BS=32 Stmin=20
:224 Byte
{~640ms}
<Up>_RxIndication(PduIdType, NotifResultType)
:FC BS=32 STmin=20
<Lo>Tp_RxIndication(PduIdType, PduInfoType)
<Lo>_TpTxConfirmation(PduIdType, NotifResultType)
<Lo>TpTxConfirmation(PduId)

Figure 3-3 Tp timeout behavior depending on the configured chunk size

3.14 AUTOSAR 4.0 CAN TP API

The APIs for the Transport Protocol modules has been redesigned in AUTOSAR 4.0. Advantages of the AUTOSAR 4.0 TP API are that the copy process of data is moved into the PduR data can be copied more efficient.

Therefore the APIs are not compatible to earlier AUTOSAR releases. A conversion mechanism provides the compatibility to Upper Layer with earlier interface implementations.

The maximum number of active parallel Upper Layer Rx and Tx API Forwardings has to be configured. Depending on the number of routing paths that shall be handled by the gateway synchronously, the maximum number has to be configured in GENy. See in chapter 6.2.1 PduR Properties Figure 6-3 PduR CAN TP 4.0 Conversion Properties.

3.15 Tp error handling

If one of the two TP components that are involved in a TP data transfer stops its transmission or reception due to an error (e.g. a Timeout) then the corresponding TP-routing relation and buffer-element will be released.

Info
There is no AUTOSAR mechanism the particular other TP component side could be notified of the error during the transmission

Therefore the next PduR_<LoTp>ProvideRxBuffer / PduR_<LoTp>CopyRxData or PduR_<LoTp>ProvideTxBuffer / PduR_<LoTp>CopyTxData of the particular other TP results in a PduR DET error with an error code " PDUR_E_UNEXPECTED_CALL " (Table 3-5 Errors reported to DET). This error could be ignored due to the return value "BUFREQ_E_NOT_OK" of PduR_<LoTp>ProvideRxBuffer/ PduR_<LoTp>CopyRxData or PduR_<LoTp>ProvideTxBuffer/ PduR_<LoTp>CopyTxData leads to a consitent release of the Tp connection. A new transmission is possible at any time.

Info
If an error occurs frequently it could be a hint to check and optimize the TP routing parameters (e.g. timeout parameter).

3.16 TP Intra ECU Communication

To realize an ECU internal diagnostic test or similar applications, the PDU Router module supports ECU internal TP I-PDU communication. For the involved modules the API behavior is similar to normal bus communication even though the PDU is not transmitted on any network.

Intra ECU communication allows a TP communication between the two upper layer modules ApplTp and DCM. Thereby the following communication scenarios are supported:

> Routing of a ApplTp Tx I-PDU to the DCM module as a DCM Rx PDU

> Routing of a DCM Tx I-PDU to the ApplTp interface as an ApplTp Rx PDU

The PDU Router supports 1:1 routing paths for intra ECU communication only. I.e. a I-PDU involved in intra ECU communication may only have a single destination PDU.

Intra ECU TP I-PDU communication is executed asynchronously by the PDU Router module. I.e. the transmission request is executed within the next call of PduR_MainFunction().

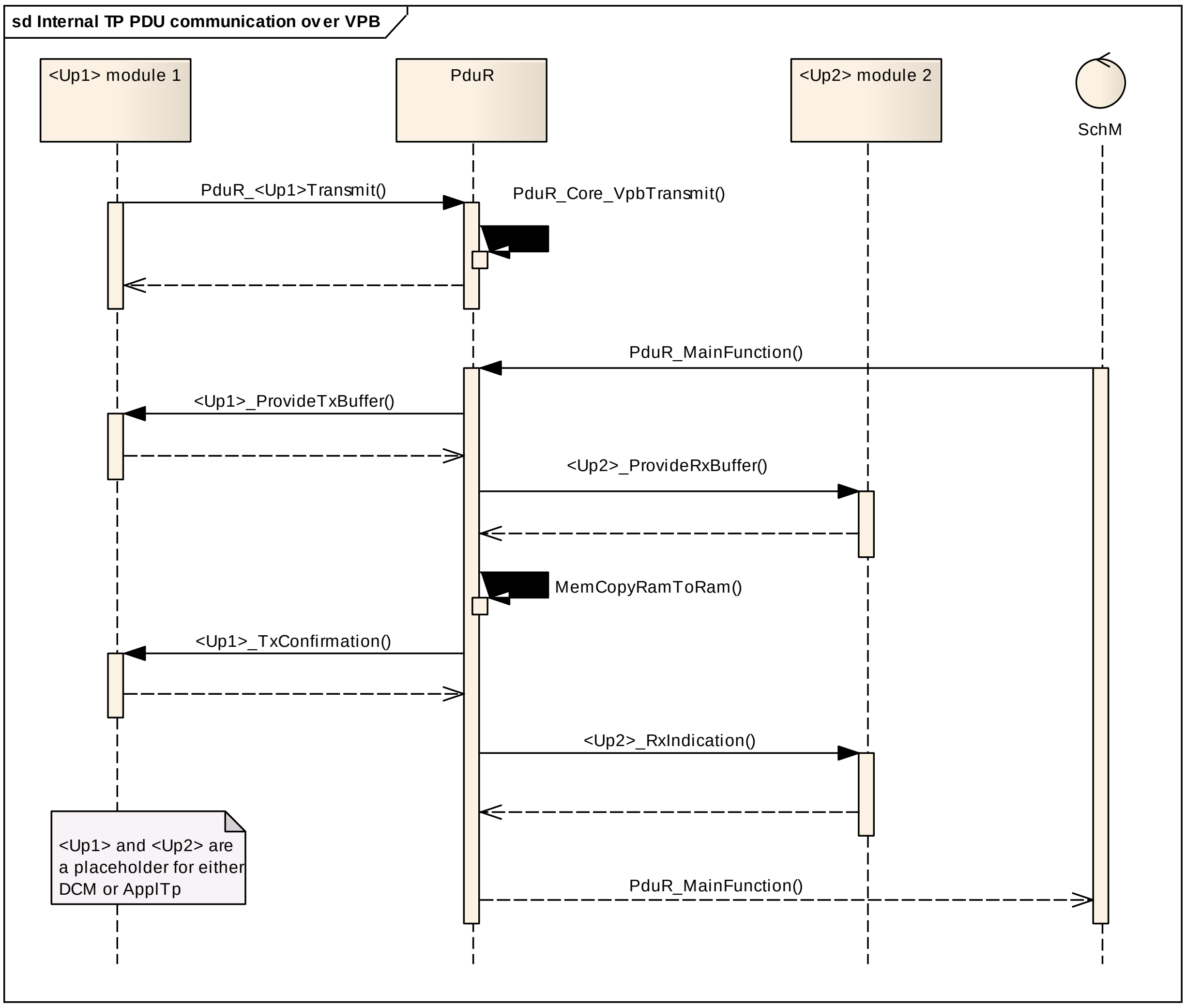


Figure 3-4 Sequence diagram of the TP intra ECU communication

As illustrated in Figure 3-4, the intra ECU PDU data is copied from the source interface (<Up1>) to the destination interface (<Up2>) in a single chunk within PduR_MainFunction().

The PDU Router requires a small state machine for any intra ECU communication that is executed. The RAM resources of such a state machine are assigned dynamically to an incoming transmission request at runtime. In case no more resources are available, the transmission request is rejected and no routing executed. The number of concurrent intra ECU communication channels is configurable at link-time.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR PDUR into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the PDUR contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
PduR.c	■		This is the source file of the PduR.
PduR.h	■	■	This is the header file of the PduR which contains global API declarations.
PduR_CanIf.h	■	■	This is the interface header of the PduR to the CAN Interface.
PduR_LinIf.h	■	■	This is the interface header of the PduR to the LIN Interface.
PduR_FrIf.h	■	■	This is the interface header of the PduR to the Flexray Interface.
PduR_MostIf.h	■	■	This is the interface header of the PduR to the MOST Interface.
PduR_CanNm.h	■	■	This is the interface header of the PduR to the CAN Network Management.
PduR_NmOsek.h	■	■	This is the interface header of the PduR to the OSEK Network Management.
PduR_FrNm.h	■	■	This is the interface header of the PduR to the Flexray Network Management.
PduR_IpduM.h	■	■	This is the interface header of the PduR to the IPDU Multiplexor.
PduR_CanTp.h	■	■	This is the interface header of the PduR to the CAN Transport Protocol.
PduR_LinTp.h	■	■	This is the interface header of the PduR to the LIN Transport Protocol.
PduR_FrTp.h	■	■	This is the interface header of the PduR to the Flexray Transport Protocol.
PduR_MostTp.h	■	■	This is the interface header of the PduR to the MOST Transport Protocol.
PduR_J1939Tp.h	■	■	This is the interface header of the PduR to the J1939 Transport Protocol.
PduR_SoAd.h	■	■	This is the interface header of the PduR to the MOST

File Name	Source Code Delivery	Object Code Delivery	Description
			Transport Protocol.
PduR_CddDobt.h	■	■	This is the interface header of the PduR to the Dobt Complex Device Driver.
PduR_Com.h	■	■	This is the interface header of the PduR to Communication.
PduR_Dcm.h	■	■	This is the interface header of the PduR to the Diagnostic Communication Manager.
PduR_ApplIf.h	■	■	This is the interface header of the PduR to the Application IF PDU Interface.
PduR_ApplTp.h	■	■	This is the interface header of the PduR to the Application Transport Protocol Interface.

Table 4-1 Static files

Info
Depending on the tailoring of the delivery, not all files might be delivered.

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool GENy.

File Name	Description
PduR_Cfg.h	This file contains pre-compile time configuration switches
PduR_Cfg.c	This file is currently not used by the PDU Router. File is empty.
PduR_Lcfg.c	This file contains link time configuration data such as allocation of RAM resources or function-pointer tables.
PduR_PBcfg.c	This file contains post-build configuration data such as routing tables.
PduR_Notifications.h	This file is not specified by AUTOSAR and is generated only if the routing callout feature has been enabled. The file contains the prototypes for the configured callout functions.
PduR_Types.h	This is the header file of the PduR which contains general types and defines.
PduR_<UL>.h	This is the interface header of the PduR to an Upper Layer Module.

Table 4-2 Generated files

4.2 Include Structure

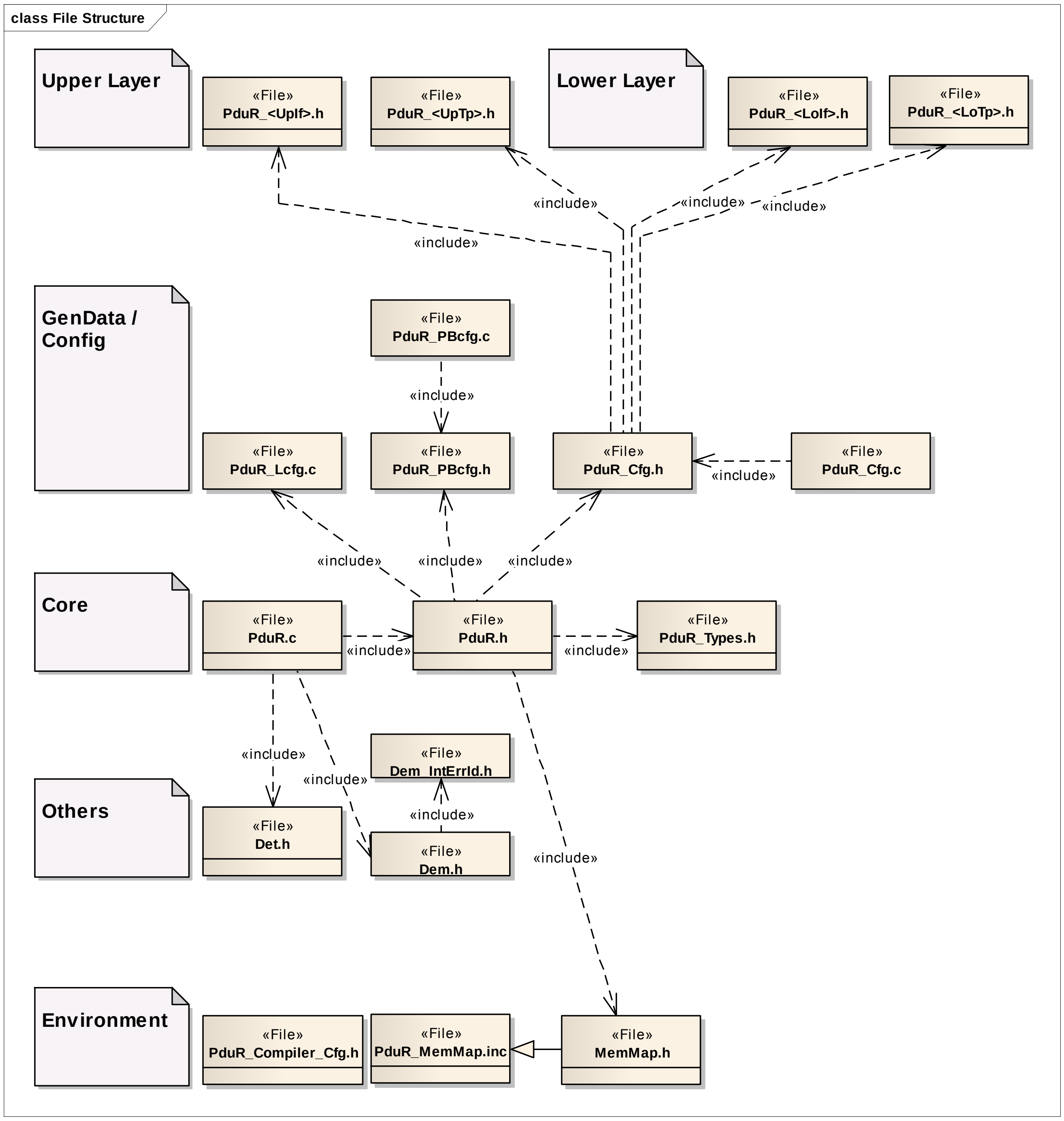


Figure 4-1 Include structure

Info
Depending on the tailoring of the delivery, the include structure may differ.

4.3 Critical Sections

The AUTOSAR standard provides with the BSW Scheduler a BSW module, which handles entering and leaving critical sections. The Vector MICROSAR PDUR supports additional solutions to handle these sections.

Critical sections are supported by the following two alternatives:

> the BSW Scheduler

> the Vector Standard Library (VStdLib)

4.4 Application Access

In order to access application PDUs the application has to include the header PduR_ApplIf.h in order to receive and transmit interface layer PDUs. The header PduR_ApplTp.h provides similar APIs for TP layer PDU access.

Info
The application access implementation assumes that the implementation of the PDU Router callback functions takes place in the same memory section as the PDU Router is mapped to. See the header files PduR_ApplIf.h and PduR_ApplTp.h for details on the used memory mapping.

5 API Description

For an interfaces overview please see Figure 2-2.

Info
Not all API functionalities might be available within your delivery. The supported functionality was derived from the questionnaire at integration time of your delivery. If required API functionality is not available in your delivery, please contact Vector Informatik.

5.1 Services provided by PDUR

5.1.1 PduR_Init

Prototype	
`void PduR_Init (const PduR_PBConfigType *ConfigPtr)`	
Parameter	
ConfigPtr	Pointer to the PDUR configuration data, if PDUR_SELECTABLE_INIT_POINTER is defined to STD_ON.
Return code	
void	none
Functional Description	
This function initializes the PDUR and performs configuration consistency checks. If the initialization is performed sucessfully the PduR is in the state PDUR_ONLINE else PDUR_UNINIT.	
Particularities and Limitations	
The function is used by the Ecu State Manager	
⚠	**Caution** PduR_Init shall not pre-empt any PDUR function.
Call Context	
The function must be called on task level and has not to be interrupted by other administrative function calls.	

Table 5-1 PduR_Init

5.1.2 PduR_InitMemory

Prototype
`void PduR_InitMemory (void)`

Parameter	
void	none
Return code	
void	none
Functional Description	
The function initialises variables, which cannot be initialised with the startup code.	
Particularities and Limitations	
The function is called by the Application.	
Call Context	
The function must be called on task level.	

Table 5-2 PduR_InitMemory

5.1.3 PduR_GetVersionInfo

Prototype	
`void` **`PduR_GetVersionInfo`** `(Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo	Pointer to where to store the version information of the PDUR.
Return code	
void	none
Functional Description	
Returns the version information of the PDUR.	
Particularities and Limitations	
The function is called by the Application.	
Call Context	
The function can be called on interrupt and task level.	

Table 5-3 PduR_GetVersionInfo

5.1.4 PduR_GetConfigurationId

Prototype	
`uint32` **`PduR_GetConfigurationId`** `(void)`	
Parameter	
void	none
Return code	
uint32	uint32
Functional Description	
Provides the unique identifier of the PDUR configuration.	

Particularities and Limitations
The function is called by the Application.
Call Context
The function can be called on interrupt and task level.

Table 5-4 PduR_GetConfigurationId

5.1.5 PduR_MainFunction

Prototype	
`void PduR_MainFunction (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
This function performs the processing of the ECU internal TP communication. The call cycle of this function affects the delay in the ECU internal TP communication.	
Particularities and Limitations	
The function is called by the BSW Scheduler.	
Call Context	
The function must be called on task level.	

Table 5-5 PduR_MainFunction

5.1.6 PduR_EnableRouting

Prototype	
`void PduR_EnableRouting (PduR_RoutingPathGroupIdType id)`	
Parameter	
id	Identification of the routing path group. Routing path groups are defined in the PDU router configuration.
Return code	
void	none
Functional Description	
This function enables a routing path group. If the routing path group does not exist or is already enabled, the function returns with no action.	
Particularities and Limitations	
The function is called by the BSW Mode Manager.	
Call Context	

The function can be called on interrupt and task level and has not to be interrupted by other PduR_EnableRouting or PduR_DisableRouting calls for the same id.

Table 5-6 PduR_EnableRouting

5.1.7 PduR_DisableRouting

Prototype	
`void` **`PduR_DisableRouting`** `(PduR_RoutingPathGroupIdType id)`	
Parameter	
id	Identification of the routing path group. Routing path groups are defined in the PDU router configuration.
Return code	
void	none
Functional Description	
This function disables a routing path group. If the routing path group does not exist or is already disbled, the function returns with no action.	
Particularities and Limitations	
The function is called by the BSW Mode Manager.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_EnableRouting or PduR_DisableRouting calls for the same id.	

Table 5-7 PduR_DisableRouting

5.1.7.1 PduR_CanTpStartOfReception

Prototype	
`BufReq_ReturnType` **`PduR_CanTpStartOfReception`** `(PduIdType CanTpRxPduId, PduLengthType TpSduLength, PduLengthType *BufferSizePtr)`	
Parameter	
CanTpRxPduId	ID of the CanTp I-PDU that will be received.
TpSduLength	Length of the entire CanTp TP SDU which will be received
BufferSizePtr	Pointer to the receive buffer in the receiving module. This parameter will be used to compute Block Size (BS) in the transport protocol module.

Return code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK Connection has been accepted. RxBufferSizePtr indicates the available receive buffer. BUFREQ_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the CanTpRxPduId is not valid or the CanTpRxPduId is not forwarded in this identity or the PduInfoPtr is not valid or the request was not accepted by the upper layer. BUFREQ_E_BUSY Currently no buffer of the requested size is available. The request was not accepted by the upper layer.
Functional Description	
This function will be called by the CanTp 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).	
Particularities and Limitations	
The function is called by CanTp.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_CanTpStartOfReception calls for the same CanTpRxPduId.	

Table 5-8 PduR_CanTpStartOfReception

5.1.8 PduR_CanTpCopyRxData

Prototype	
`BufReq_ReturnType` **`PduR_CanTpCopyRxData`** `(PduIdType CanTpRxPduId, PduInfoType *PduInfoPtr, PduLengthType *BufferSizePtr)`	
Parameter	
CanTpRxPduId	ID of the CanTp I-PDU that will be received.
PduInfoPtr	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.
BufferSizePtr	Available receive buffer after data has been copied.

Return code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK Buffer request accomplished successful. BUFREQ_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the CanTpRxPduId is not valid or the CanTpRxPduId is not forwarded in this identity or the PduInfoPtr is not valid or the request was not accepted by the upper layer. BUFREQ_E_BUSY Currently no buffer of the requested size is available. It's up the requestor to retry request for a certain time. The request was not accepted by the upper layer. BUFREQ_E_OVFL The upper TP module is not able to receive the number of bytes. The request was not accepted by the upper layer.
Functional Description	
This function is called by the CanTp if data has to be to copied to the receiving module. Several calls may be made during one transportation of an I-PDU.	
Particularities and Limitations	
The function is called by CanTp.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_CanTpCopyRxData calls for the same CanTpRxPduId.	

Table 5-9 PduR_CanTpCopyRxData

5.1.9 PduR_CanTpCopyTxData

Prototype	
`BufReq_ReturnType` **`PduR_CanTpCopyTxData`** `(PduIdType CanTpTxPduId, PduInfoType *PduInfoPtr, RetryInfoType *RetryInfoPtr, PduLengthType *TxDataCntPtr)`	
Parameter	
CanTpTxPduId	ID of the CanTp I-PDU that will be transmitted.
PduInfoPtr	Pointer to the 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 no enough transmit data is available, no data is copied. The transport protocol module will retry. A size of copy size of 0 can be used to indicate state changes in the retry parameter.

RetryInfoPtr	This parameter is used to retransmit data because problems occurred in transmitting it the last time. If the I-PDU is transmitted from a local module 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 can be provided. Alternatively TpDataState can indicate TP_NORETRY. Both indicate 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.
TxDataCntPtr	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 code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK The data has been copied to the transmit buffer successful. BUFREQ_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the CanTpTxPduId is not valid or the CanTpTxPduId is not forwarded in this identity or the PduInfoPtr is not valid or the request was not accepted by the upper layer and no data has been copied. BUFREQ_E_BUSY The request cannot be processed because the TX data is not available and no data has been copied. The TP layer might retry later the copy process.
Functional Description	
This function is called by the CanTp 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.	
Particularities and Limitations	
The function is called by CanTp.	
Call Context	

The function can be called on interrupt and task level and has not to be interrupted by other PduR_CanTpCopyTxData calls for the same CanTpTxPduId.

Table 5-10 PduR_CanTpCopyTxData

5.1.10 PduR_CddDobtSetCanTpState

Prototype	
`void` **`PduR_CddDobtSetCanTpState`** `(boolean CddDobtState)`	
Parameter	
CddDobtState	TRUE: CddDobt is active. FALSE: CddDobt is not active.
Return code	
void	none
Functional Description	
The function is called by CddDobt to switch configured CanTp calls API to CddDobt forwardings.	
Particularities and Limitations	
The function is called by CddDobt.	
Call Context	
The must be called in the context of CddDobt_PduRCanTpCallout function.	

Table 5-11 PduR_CddDobtSetCanTpState

5.1.11 Provided Interfaces to Lower Communication Interface Layers

This chapter describes the interfaces provided to Lower Layers using the Communication Interface (e.g. CanIf, LinIf, FrIf). Replace the tag <LoIf> by the MSN of the Upper Layer.

5.1.11.1 PduR_<LoIf>RxIndication

Prototype	
`void` **`PduR_<LoIf>RxIndication`** `(PduIdType <LoIf>RxPduId, PDUR_RXINDICATION_TYPE PDUR_RXINDICATION_PARA)`	
Parameter	
<LoIf>RxPduId	Handle ID of the received <LoIf> I-PDU.
PDUR_RXINDICATION_PARA	SduPtr Pointer to the received I-PDU data. This Parameter is used, if PDUR_USE_PDUINFOTYPE is defined to STD_OFF. PduInfoPtr Payload information of the received I-PDU (pointer to data and data length). This Parameter is used, if PDUR_USE_PDUINFOTYPE is defined to STD_ON.
Return code	
void	none

Functional Description
The function is called by the <LoIf> to indicate the complete reception of a <LoIf> I-PDU. The PDU Router evaluates the <LoIf> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper IF module using the appropriate I-PDU handle of the destination layer.
Particularities and Limitations
The function is called by <LoIf>.
Call Context
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoIf>RxIndication calls for the same <LoIf>RxPduId.

Table 5-12 PduR_<LoIf>RxIndication

5.1.11.2 PduR_<LoIf>TxConfirmation

Prototype	
`void PduR_<LoIf>TxConfirmation (PduIdType <LoIf>TxPduId)`	
Parameter	
<LoIf>TxPduId	Handle ID of the transmitted <LoIf> I-PDU.
Return code	
void	none
Functional Description	
The function is called by the <LoIf> to indicate the complete transmission of a <LoIf> I-PDU. The PDU Router evaluates the <LoIf> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper IF module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoIf>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoIf>TxConfirmation calls for the same <LoIf>TxPduId.	

Table 5-13 PduR_<LoIf>TxConfirmation

5.1.11.3 PduR_<LoIf>TriggerTransmit

Prototype	
`PDUR_TRIGGERTRANSMIT_RETURN PduR_<LoIf>TriggerTransmit (PduIdType <LoIf>TxPduId, PDUR_TRIGGERTRANSMIT_TYPE PDUR_TRIGGERTRANSMIT_PARA)`	
Parameter	
<LoIf>TxPduId	Handle ID of the <LoIf> I-PDU that will be transmitted.

PDUR_TRIGGERTRANSMIT_PARA	SduPtr Pointer to the I-PDU data buffer of the <LoIf> IF that will be transmitted. This Parameter is used, if PDUR_USE_PDUINFOTYPE is defined to STD_OFF. PduInfoPtr Payload information of the <LoIf> I-PDU that will be transmitted (pointer to data and data length). This Parameter is used, if PDUR_USE_PDUINFOTYPE is defined to STD_ON.
Return code	
PDUR_TRIGGERTRANSMIT_RETURN	PDUR_TRIGGERTRANSMIT_RETURN none No return value. This return value is used, if PDUR_USE_PDUINFOTYPE is defined to STD_OFF. Std_ReturnType This return value is used, if PDUR_USE_PDUINFOTYPE is defined to STD_ON. E_OK: The SDU has been copied and the SduLength indicates the number of copied bytes. E_NOT_OK: The SDU has not been copied and the SduLength has not been set.
Functional Description	
The function is called by the <LoIf> to request the <LoIf> TX I-PDU data before transmission. The PDU Router evaluates the <LoIf> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper IF module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoIf>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoIf>TriggerTransmit calls for the same <LoIf>TxPduId.	

Table 5-14 PduR_<LoIf>TriggerTransmit

5.1.12 Provided Interfaces to Lower Transport Protocol Layers

This chapter describes the interfaces provided to Lower Layers using the Transport Protocol (e.g. CanTp, LinTp, FrTp). Replace the tag <LoTp> by the MSN of the Upper Layer.

5.1.12.1 PduR_<LoTp>ProvideRxBuffer

Prototype	
`BufReq_ReturnType` **`PduR_<LoTp>ProvideRxBuffer`** `(PduIdType <LoTp>RxPduId, PduLengthType TpSduLength, PduInfoTypeApplPtr *PduInfoPtr)`	
Parameter	
<LoTp>RxPduId	ID of the <LoTp> I-PDU that will be received.
TpSduLength	Length of the entire <LoTp> TP SDU which will be received
PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a receive buffer. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.

Return code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK Buffer request accomplished successful BUFREQ_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <LoTp>TxPduId is not valid or the <LoTp>TxPduId is not forwarded in this identity or the PduInfoPtr is not valid or the request was not accepted by the destination upper layer. BUFREQ_E_BUSY Currently no buffer available BUFREQ_E_OVFL The upper TP module is not able to receive number of TpSduLength bytes and no buffer is provided.
Functional Description	
The function is called by the <LoTp> to receive a <LoTp> SDU. The PDU Router evaluates the <LoTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoTp>ProvideRxBuffer calls for the same <LoTp>RxPduId.	

Table 5-15 PduR_<LoTp>ProvideRxBuffer

5.1.12.2 PduR_<LoTp>RxIndication

Prototype	
void **PduR_<LoTp>RxIndication** (PduIdType <LoTp>RxPduId, NotifResultType Result)	
Parameter	
<LoTp>RxPduId	ID of the <LoTp> I-PDU that will be received.

Result	Result of the TP reception NTFRSLT_OK The TP reception has been completed successfully. NTFRSLT_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <LoTp>RxPduId is not valid or the <LoTp>RxPduId is not forwarded in this identity or the request was not accepted by the destination upper layer. NTFRSLT_E_TIMEOUT_A The TP reception has not been completed successfully. NTFRSLT_E_TIMEOUT_Cr The TP reception has not been completed successfully. NTFRSLT_E_WRONG_SN The TP reception has not been completed successfully. NTFRSLT_E_UNEXP_PDU The TP reception has not been completed successfully. NTFRSLT_E_NO_BUFFER The TP reception has not been completed successfully. NTFRSLT_E_CANCELLATION_OK The TP reception has been cancelled successfully. NTFRSLT_E_CANCELLATION_NOT_OK The TP reception has not been cancelled.
Return code	
void	none
Functional Description	
The function is called by the <LoTp> to indicate the complete receiption of a <LoTp> SDU or to report an error that occurred during reception. The PDU Router evaluates the <LoTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoTp>RxIndication calls for the same <LoTp>RxPduId.	

Table 5-16 PduR_<LoTp>RxIndication

5.1.12.3 PduR_<LoTp>ProvideTxBuffer

Prototype	
`BufReq_ReturnType` **`PduR_<LoTp>ProvideTxBuffer`** `(PduIdType <LoTp>TxPduId, PduInfoTypeApplPtr *PduInfoPtr, uint16 Length)`	
Parameter	
<LoTp>TxPduId	ID of the <LoTp> I-PDU that will be transmitted.

PduInfoPtr	Pointer to pointer to PduInfoStructure containing SDU data pointer and SDU length of a transmit buffer. If the return value is not equal to BUFREQ_OK, PduInfoPtr is undefined and shall not be used.
Length	Exact length of the requested transmit buffer. It shall not exceed the number of bytes still to be sent. This parameter is needed by the transport protocol to perform error recovery mechanisms. If no error recovery is configured for this PduId, Length may be zero, which indicates that the provided buffer can be of arbitrary size (larger than zero).
Return code	
BufReq_ReturnType	BufReq_ReturnType BUFREQ_OK Buffer request accomplished successful. BUFREQ_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <LoTp>TxPduId is not valid or the <LoTp>TxPduId is not forwarded in this identity or the PduInfoPtr is not valid or the request was not accepted by the destination upper layer. BUFREQ_E_BUSY Currently no buffer of the requested size is available
Functional Description	
The function is called by the <LoTp> to request a <LoTp> TX SDU before transmission. The called module has to copy the data to be transmitted to the destination of PduInfoPtr. The PDU Router evaluates the <LoTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoTp>ProvideTxBuffer calls for the same <LoTp>TxPduId.	

Table 5-17 PduR_<LoTp>ProvideTxBuffer

5.1.12.4 PduR_<LoTp>TxConfirmation

Prototype	
`void` **`PduR_<LoTp>TxConfirmation`** `(PduIdType <LoTp>TxPduId, NotifResultType Result)`	
Parameter	
<LoTp>TxPduId	ID of the <LoTp> I-PDU that will be transmitted.

Result	Result of the TP transmission NTFRSLT_OK The TP transmission has been completed successfully. NTFRSLT_E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <LoTp>TxPduId is not valid or the <LoTp>TxPduId is not forwarded in this identity or the request was not accepted by the destination upper layer. NTFRSLT_E_TIMEOUT_A The TP transmission has not been completed successfully. NTFRSLT_E_TIMEOUT_Bs The TP transmission has not been completed successfully. NTFRSLT_E_INVALID_FS The TP transmission has not been completed successfully. NTFRSLT_E_WFT_OVRN The TP transmission has not been completed successfully. NTFRSLT_E_NO_BUFFER The TP transmission has not been completed successfully. NTFRSLT_E_CANCELLATION_OK The TP transmission has been cancelled successfully. NTFRSLT_E_CANCELLATION_NOT_OK The TP transmission has not been cancelled.
Return code	
void	none
Functional Description	
The function is called by the <LoTp> to confirm a successful transmission of a <LoTp> TX SDU or to report an error that occurred during transmission. The PDU Router evaluates the <LoTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoTp>TxConfirmation calls for the same <LoTp>TxPduId.	

Table 5-18 PduR_<LoTp>TxConfirmation

5.1.12.5 PduR_<LoTp>ChangeParameterConfirmation

Prototype	
`void` **`PduR_<LoTp>ChangeParameterConfirmation`** `(PduIdType <LoTp>RxPduId, NotifResultType Result)`	
Parameter	
<LoTp>RxPduId	ID of the <LoTp> I-PDU that parameter has been changed.

Result	Result of the TP parameter change NTFRSLT_PARAMETER_OK The parameter change has been executed successfully. NTFRSLT_E_RX_ON The parameter change request has not been executed successfully due to an ongoing reception. NTFRSLT_E_VALUE_NOT_OK The parameter change request has not been executed successfully due to a wrong value. NTFRSLT_E_PARAMETER_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <LoTp>RxPduId is not valid or the <LoTp>TxPduId is not forwarded in this identity or the parameter change was not accepted by the destination lower layer.
Return code	
void	none
Functional Description	
The function is called by the <LoTp> to confirm a successful parameter change or to report an error that occurred during transmission. The PDU Router evaluates the <LoTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to an upper TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <LoTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<LoTp>TxConfirmation calls for the same <LoTp>TxPduId.	

Table 5-19 PduR_<LoTp>ChangeParameterConfirmation

5.1.13 Provided Interfaces to Upper Layers

This chapter describes the interfaces provided to Upper Layers (e.g. Com, Dcm, IpduM, ApplIf, ApplTp and Cdds). Replace the tag <Up> by the MSN of the Upper Layer.

5.1.13.1 PduR_<Up>Transmit

Prototype	
`Std_ReturnType` **`PduR_<Up>Transmit`** `(PduIdType ComTxPduId, const PduInfoType *PduInfoPtr)`	
Parameter	
<Up>TxPduId	ID of the <Up> I-PDU to be transmitted
PduInfoPtr	Payload information of the I-PDU (pointer to data and data length)

Return code	
Std_ReturnType	Std_ReturnType E_OK The request was accepted by the PDU Router and by the destination layer. E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <Up>TxPduId is not valid or the <Up>TxPduId is not forwarded in this identity or the PduInfoPtr is not valid or the request was not accepted by the destination layer.
Functional Description	
The function serves to request the transmission of an IF or TP layer I-PDU. The PDU Router evaluates the <Up> I-PDU handle and performs appropriate handle and port conversion. The call is routed to a lower IF module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <Up>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<Up>Transmit calls for the same <Up>TxPduId.	

Table 5-20 PduR_<Up>Transmit

5.1.14 Provided Interfaces to Upper Transport Protocol Layers

This chapter describes the interfaces provided to Upper Layers (e.g. Dcm, ApplTp and Cdds). Replace the tag <UpTp> by the MSN of the Upper Layer.

5.1.14.1 PduR_<UpTp>ChangeParameterRequest

Prototype	
`Std_ReturnType` **`PduR_<UpTp>ChangeParameterRequest`** `(PduIdType <UpTp>RxPduId, TPParameterType parameter, uint16 value)`	
Parameter	
<UpTp>RxPduId	ID of the <UpTp> I-PDU where the parameter has to be changed
parameter	The TP parameter that shall be changed.
value	The new value for the TP parameter.
Return code	
Std_ReturnType	Std_ReturnType E_OK The request was accepted by the PDU Router and by the destination layer. E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <UpTp>TxPduId is not valid or the <UpTp>TxPduId is not forwarded in this identity or the request was not accepted by the TP layer.

Functional Description
The function serves to change a specific transport protocol parameter (e.g. block-size). The PDU Router evaluates the <UpTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to a lower TP module using the appropriate I-PDU handle of the destination layer.
Particularities and Limitations
The function is called by <UpTp>.
Call Context
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<UpTp>ChangeParameterRequest calls for the same <UpTp>RxPduId.

Table 5-21 PduR_<UpTp>ChangeParameterRequest

5.1.14.2 PduR_<UpTp>ReadParameterRequest

Prototype	
`Std_ReturnType` **`PduR_<UpTp>ReadParameterRequest`** `(PduIdType <UpTp>RxPduId, TPParameterType parameter, uint16 *valuePtr)`	
Parameter	
<UpTp>RxPduId	ID of the <UpTp> I-PDU where the parameter has to be changed
parameter	The TP parameter that shall be read.
valuePtr	The pointer to buffer to return the parameter value.
Return code	
Std_ReturnType	Std_ReturnType E_OK The request was accepted by the PDU Router and by the destination layer. E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <UpTp>TxPduId is not valid or the <UpTp>TxPduId is not forwarded in this identity or the request was not accepted by the TP layer.
Functional Description	
The function serves to read a specific transport protocol parameter (e.g. block-size). The PDU Router evaluates the <UpTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to a lower TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <UpTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<UpTp>ReadParameterRequest calls for the same <UpTp>RxPduId.	

Table 5-22 PduR_<UpTp>ReadParameterRequest

5.1.14.3 PduR_<UpTp>CancelReceiveRequest

Prototype	
`Std_ReturnType` **`PduR_<UpTp>CancelReceiveRequest`** `(PduIdType <UpTp>RxPduId)`	
Parameter	
<UpTp>RxPduId	ID of the RX <UpTp> I-PDU to be cancelled
Return code	
Std_ReturnType	Std_ReturnType E_OK The cancellation request was accepted by the PDU Router and by the TP layer. E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <UpTp>RxPduId is not valid or the <UpTp>RxPduId is not forwarded in this identity or the request was not accepted by the TP layer.
Functional Description	
The function serves to cancel the reception of a TP layer I-PDU. The PDU Router evaluates the <UpTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to a lower TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <UpTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<UpTp>CancelReceiveRequest calls for the same <UpTp>RxPduId.	

Table 5-23 PduR_<UpTp>CancelReceiveRequest

5.1.14.4 PduR_<UpTp>CancelTransmitRequest

Prototype	
`Std_ReturnType` **`PduR_<UpTp>CancelTransmitRequest`** `(PduIdType <UpTp>TxPduId)`	
Parameter	
<UpTp>TxPduId	ID of the TX <UpTp> I-PDU to be cancelled

Return code	
Std_ReturnType	Std_ReturnType E_OK The cancellation request was accepted by the PDU Router and by the TP layer. E_NOT_OK The PDU Router is in the PDUR_UNINIT state or the <UpTp>TxPduId is not valid or the <UpTp>TxPduId is not forwarded in this identity or the request was not accepted by the TP layer.
Functional Description	
The function serves to cancel the transmission of a TP layer I-PDU. The PDU Router evaluates the <UpTp> I-PDU handle and performs appropriate handle and port conversion. The call is routed to a lower TP module using the appropriate I-PDU handle of the destination layer.	
Particularities and Limitations	
The function is called by <UpTp>.	
Call Context	
The function can be called on interrupt and task level and has not to be interrupted by other PduR_<UpTp>CancelTransmitRequest calls for the same <UpTp>TxPduId.	

Table 5-24 PduR_<UpTp>CancelTransmitRequest

5.1.15 [Service Name]

5.2 Services used by PDUR

In the following table services provided by other components, which are used by the PDUR are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError
DEM	Dem_ReportErrorStatus
EcuM	EcuM_GeneratorCompatibilityError
CddDobt	CddDobt_Transmit CddDobt_PduRCanTpCallout
Upper Layer with Communication Interface (e.g. COM, Dcm, IpduM, ApplIf, Cdds)	<Up>_RxIndication <Up>_TxConfirmation <Up>_TriggerTransmit
Upper Layer with Transport Protocol Interface (e.g. COM, Dcm, ApplTp, Cdds)	<Up>_TpRxIndication <Up>_TpProvideRxBuffer <Up>_TpProvideTxBuffer <Up>_TpTxConfirmation <Up>_ChangeParameterConfirmation *

Component	API
Lower Layer with Communication Interface (e.g. CanIf, LinIf, FrIf, IpduM, CanNm)	<Lo>_Transmit
Lower Layer with Transport Protocol Interface (e.g. CanTp, LinTp, FrTp)	<Lo>_Transmit <Lo>_ChangeParameterRequest * <Lo>_ReadParameterRequest * <Lo>_CancelReceiveRequest * <Lo>_CancelTransmitRequest *
SchM	SchM_Enter_PduR SchM_Exit_PduR

Table 5-25 Services used by the PDUR

* The interface is not available for every layer.

5.3 Dependencies to other BSW Modules used by PDUR

To allow reliable gateway operation, the following module settings are required by the PDU Router. The configuration is executed automatically by the PDU Router where ever this is possible.

Component	Description
CanIf	**Transmit Buffer:** To avoid loss of I-PDUs due to an occupied Tx hardware object the CAN IF must use an own Tx buffer for each I-PDU.
LinIf	**Enable PduR Confirmation:** In case of FiFo queued routing paths, the PDU Router relays on the confirmation of any transmitted I-PDU.
FrIf	**Immediate Tx:** This transmission mode can only be used if the transmission request is synchronized to the FlexRay bus time. As the time of routing (which is triggered by the reception of an I-PDU) is not guaranteed to be synchronous with the FlexRay scheduling, this option may not be used and is disabled by GENy for all Tx I-PDUs of a routing path.
Com	**COM Generate:** This COM Attribute specifies if the I-PDU is handed by COM or not. If the I-PDU attribute is not checked, no signal access and other COM features will be provided by COM. For details, see the TechnicalReference_Il_AsrCom.pdf. **Tx COM I-PDU:** Tx I-PDUs that are part of a routing path must not be handled by COM. As there must be only a single transmitting BSW module for any I-PDU. The COM Generate is disabled automatically when a Tx I-PDU is added to a routing path. **Rx COM I-PDU:** Rx I-PDUs that are part of a routing path can also be handled by COM. In order to optimize runtime of the communication stack (and especially of the Rx indication), COM Generate can be activated manually for Rx I-PDUs that are not used by the application.

5.3.1 Callout Functions

At its configurable interfaces the PDUR defines callout functions. The declarations of the callout functions are provided by the BSW module, i.e. the PDUR. It is the integrator's task to provide the corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs. The PDUR callout function declarations are described in the following tables:

5.3.2 IF Routing Callout

Prototype	
`boolean` **`<Configured IF Routing Callout>`** `(PduInfoType *pPduInfo )`	
Parameter	
`pPduInfo`	Structure containing a pointer to the data that will be routed (SduDataPtr) and the length of the data that will be routed (SduLength).
Return code	
`boolean`	TRUE if the routing shall be processed, else FALSE.
Functional Description	
This feature is an extension of the AUTOSAR 3.0 feature set. Its functionality has been derived from Bugzilla #18729 “RFC – Additional Callout to control Pdu Gatewayed by PduR”. The gateway callout function allows controlling PDU gateway activity. Further more that API can be used to set internal ECU states based on routing activity.	
Particularities and Limitations	
> The Feature must be enabled at pre-compile time. > The Callout functions can be added and removed at link time. > It is not supported to remove callout functions at post-build time > This function is asynchronous. > This function is reentrant for different pPduInfo handles.	
Expected Caller Context	
> The function can be called on interrupt and task level and is not to be interrupted by other calls for the same pPduInfo parameter.	

Table 5-26 IF Routing Callout

6 Configuration

In the PDUR the attributes can be configured according to/ with the following methods/ tools:

> Configuration in GENy for a detailed description see 6.1, 6.2, 6.3, 6.4

> Configuration in EcuC Data Base for a detailed description see 6.5

6.1 Configuration Variants

The PDUR supports the configuration variants

> `VARIANT-LINK-TIME`

> `VARIANT-POST-BUILD`

The configuration classes of the PDUR parameters depend on the supported configuration variants. For their definitions please see the PduR_bswmd.arxml file.

6.2 Configuration with GENy

The PDUR is configured with the help of the configuration tool GENy.

Three types of routing paths are supported by PDU Router, API forwarding, Interface layer- and TP layer routing paths. API forwarding routing paths is connecting a lower-layer I-PDU with an I-PDU of an upper layer and vice versa (e.g. the forwarding of a CAN IF I-PDU to COM). These routing paths are used for stack internal communication only.

Interface and TP layer routing paths can be observed externally as they route incoming I-PDUs to one or several external I-PDUs.

Info
If the PDU Router module is intended to be used with BSW modules from other vendors please contact Vector Informatik.

6.2.1 PduR Properties

Depending on the configuration variant and the tailoring made by Vector Informatik at delivery time not all settings are configurable or visible.

If the TP and/or interface layer gateway have not been licensed, the tree elements for routing paths and TP buffer elements are displayed anyway. When right clicking on one of the elements a menu is illustrated that seems to allow adding related elements. However, if the requested feature is not active the request to add such an element is of no effect.

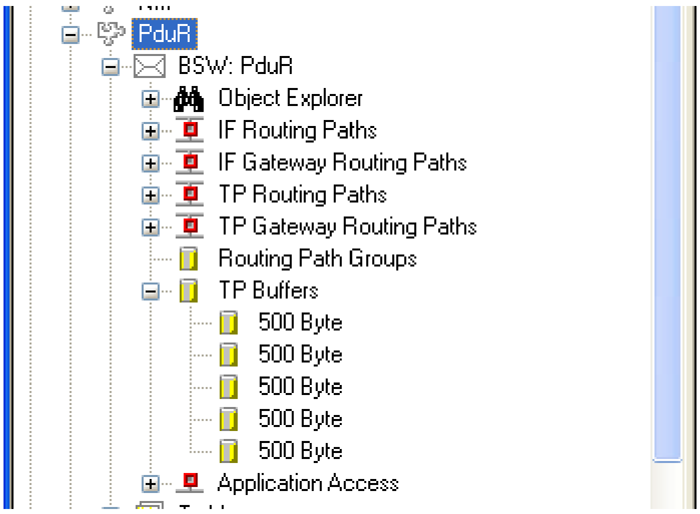
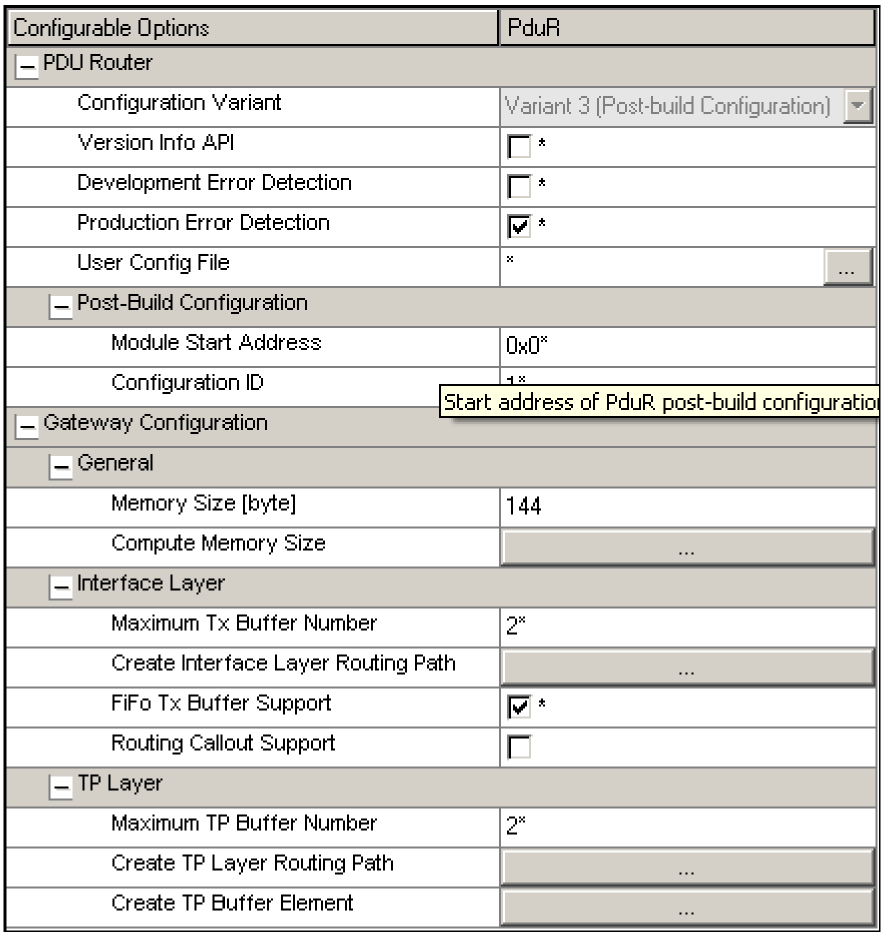

PduR
BSW: PduR
Object Explorer
IF Routing Paths
IF Gateway Routing Paths
TP Routing Paths
TP Gateway Routing Paths
Routing Path Groups
TP Buffers
500 Byte
500 Byte
500 Byte
500 Byte
500 Byte
Application Access

Figure 6-1 PduR Configuration



Configurable Options	PduR
PDU Router	
Configuration Variant	Variant 3 (Post-build Configuration)
Version Info API	☐ *
Development Error Detection	☐ *
Production Error Detection	✓ *
User Config File	* ...
Post-Build Configuration	
Module Start Address	0x0*
Configuration ID	1*
Gateway Configuration	
General	
Memory Size [byte]	144
Compute Memory Size	...
Interface Layer	
Maximum Tx Buffer Number	2*
Create Interface Layer Routing Path	...
FiFo Tx Buffer Support	✓ *
Routing Callout Support	☐
TP Layer	
Maximum TP Buffer Number	2*
Create TP Layer Routing Path	...
Create TP Buffer Element	...

Start address of PduR post-build configuratio

Figure 6-2 PduR Parameters

Attribute Name	Value Type	Values The default value is written in bold	Description
Configuration Variant	Enum		Supported configuration variant of the PDU Router delivery. This setting is pre-configured by Vector Informatik: 2) Link-Time Configuration: Similar to the AUTOSAR post-build time configuration but optimized regarding runtime and usability. Configuration tables are linked to the code directly without requiring indirection over the post-build configuration structure. This configuration variant does not support post-build time configuration. 3) Post-Build Time Configuration: This variant allows post-build configuration of parameters generated to PduR_PBcfg.c.
Version Info API	Boolean		If the checkbox is enabled, the PDU Router provides an API to retrieve the PDU Router version by the service function PduR_GetVersionInfo().
Development Error Detection	Boolean		If the checkbox is enabled, the PDU Router performs runtime consistency checks that can help to detect development errors. Errors are reported to the Development Error Tracer (DET). Typically this error checking is not included in production code.
Production Error Detection	Boolean		If the checkbox is enabled, the PDU Router performs runtime checks for critical errors. Detected errors are reported to the Diagnostics Event Manager (DEM). These error checks are required by AUTOSAR for development and production code and are only disabled upon customer request.
User Config File	String		The content of the user configuration file is appended to the generated PduR_Cfg.h file. By means of the user configuration file it is possible to modify or extend the PDU Router configuration beyond the generator capabilities.
Module Start Address	String		This attribute is relevant for configuration at post-build time only. The attribute indicates the start address of the post-build configuration structure within ROM. The address is used by the post-build generator to compute absolute addresses required to access referenced configuration tables.
Configuration ID	Integer		The configuration ID is only relevant for configurations with post-build capability and can

Attribute Name	Value Type	Values The default value is written in bold	Description
			be configured at link and post-build time to a user specific value that allows identifying the current PDU Router configuration by the API PduR_GetVersionInfo().
Memory Size [byte]	Integer		This link-time parameter specifies how much RAM the PDU Router module will allocate for its PDU buffers. The size must be at least of the size required by the current link-time configuration. Additional RAM might be allocated to allow further RAM consumption at post-build time. See chapter 3.6 for details. Please not that the configured number does not specify the complete RAM consumption of PDU Router as the module uses further memory for global variables. Please note that the memory size does not indicate the over all RAM consumption of the PDU Router.
Compute Memory Size	Action		This button allows computing the current memory size that is used by the PDU Router module buffer. The value will be written to the “Memory Size [byte]” Attribute.
Maximum Tx Buffer Number	Integer		If FiFo queues are used, the maximum number of FiFo queues must be configured to allow adding further FiFo queues at post-build configuration time.
Create Interface Layer Routing Path	Action		Creates a manual (user defined) interface layer routing path.
FiFo Tx Buffer Support			This switch is available in GENy for deliveries supporting library and post-build time configuration. The switch specifies whether the gateway shall support FiFo queued routing or not.
Routing Callout Support	Boolean		If set, the PDU Router module provides IF routing callouts.
Maximum TP Buffer Number	Integer		Allows the user to specify the maximum number of TP buffer elements that are configurable at post-build time. The value must not be smaller that the number configured at link time.
Create TP Layer Routing Path	Action		Creates a manual (user defined) TP layer routing path.
Create TP Buffer Element	Action		Creates a TP buffer element as required by TP layer routing paths.

Table 6-1 PduR Properties

Attribute Name	Value Type	Values The default value is written in bold	Description
Maximum Tx ApplTp connections via CanTp	Integer		Specifies the maximum number of concurrent CanTp Tx connection that can be hold by ApplTp. Only connections transmitted via CanTp have to be considered.
Maximum Tx COM connections via CanTp	Integer		Specifies the maximum number of concurrent CanTp Tx connection that can be hold by Com. Only connections transmitted via CanTp have to be considered.
Maximum Tx DCM connections via CanTp	Integer		Specifies the maximum number of concurrent CanTp Tx connection that can be hold by Dcm. Only connections transmitted via CanTp have to be considered.
Maximum Rx CanTp connections used by upper layers	Integer		Specifies the maximum number of concurrent CanTp Rx connections that can be forwarded to an upper layer (i.e. DCM, COM, ApplTp). The value excludes CanTp connections used for TP layer routing relations. The configuration is required for the API style conversion that is carried out as the upper layer modules still uses the AUTOSAR 3.x compliant API.

Figure 6-3 PduR CAN TP 4.0 Conversion Properties

6.2.2 Automatic Routing Path Configuration

6.2.2.1 Interface and Transport Protocol API Forwarding

All API forwarding routing information is automatically derived from upper- (i.e. COM and DCM) and lower- layers (e.g. CAN IF or FR TP) using the GENy framework. The same applies for the TP intra ECU communication.

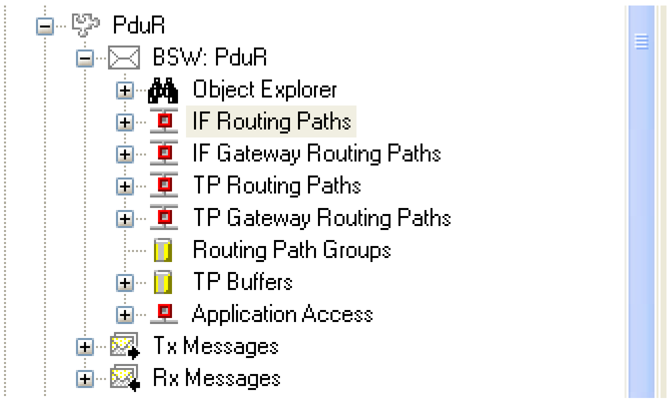
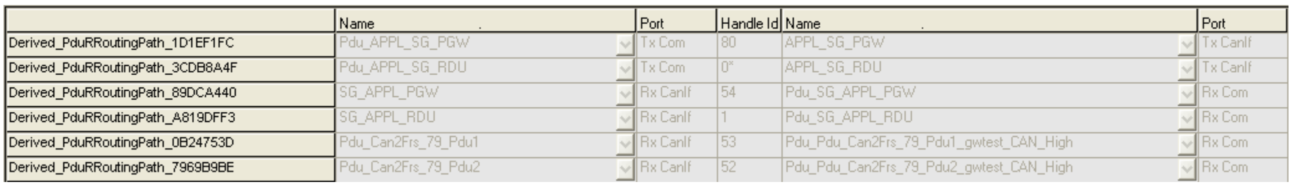


Figure 6-4 IF and TP Routing Paths visualize API Forwardings



	Name	Port	Handle Id	Name	Port
Derived_PduRRoutingPath_1D1EF1FC	Pdu_APPL_SG_PGW	Tx Com	80	APPL_SG_PGW	Tx CanIf
Derived_PduRRoutingPath_3CDB8A4F	Pdu_APPL_SG_RDU	Tx Com	0*	APPL_SG_RDU	Tx CanIf
Derived_PduRRoutingPath_89DCA440	SG_APPL_PGW	Rx CanIf	54	Pdu_SG_APPL_PGW	Rx Com
Derived_PduRRoutingPath_A819DFF3	SG_APPL_RDU	Rx CanIf	1	Pdu_SG_APPL_RDU	Rx Com
Derived_PduRRoutingPath_0B24753D	Pdu_Can2Frs_79_Pdu1	Rx CanIf	53	Pdu_Pdu_Can2Frs_79_Pdu1_gwtest_CAN_High	Rx Com
Derived_PduRRoutingPath_7969B9BE	Pdu_Can2Frs_79_Pdu2	Rx CanIf	52	Pdu_Pdu_Can2Frs_79_Pdu2_gwtest_CAN_High	Rx Com

Figure 6-5 API Forwarding Parameters

6.2.2.2 Interface Gateway Routing Paths

Depending on OEM requirements GENy can detect and configure interface layer routing paths automatically. The rules for finding and creating routing paths are OEM dependent (i.e. according to similar CAN IDs in a certain range...).

Automatic routing paths do not allow modification of the source and destination I-PDU and cannot be deleted from the configuration. Please note that deletion of automatic routing paths is not possible even as the “Remove routing path” menu is illustrated.

Automatic routings relation can, however, be disabled by clearing the “Generate” flag.

Edit
If not pre-configured by OEM requirements, the memory size of each interface layer routing path has to be configured manually.

Re- check all routing path settings after a database update as the update can cause creation of new routing paths.

6.2.2.3 Transport Protocol Gateway Routing Paths

Depending on OEM requirements GENy can detect TP layer routing paths automatically. The rules for finding and creating routing paths are OEM dependent (i.e. according to similar CAN IDs in a certain range...).

Automatic routing paths do not allow modification of the source and destination I-PDU and cannot be deleted from the configuration. Please note that deletion of automatic routing paths is not possible even as the “Remove routing path” menu is illustrated.

Automatic routings relation can, however, be disabled by clearing the “Generate” flag.

6.2.3 Manual Gateway Routing Path Configuration

6.2.3.1 Creating a routing path

To create an IF or TP routing path right click on the “IF Gateway Routing Paths” or “TP Gateway Routing Paths” item in the GENy modules tree and select “Add new Gateway Routing Path”.

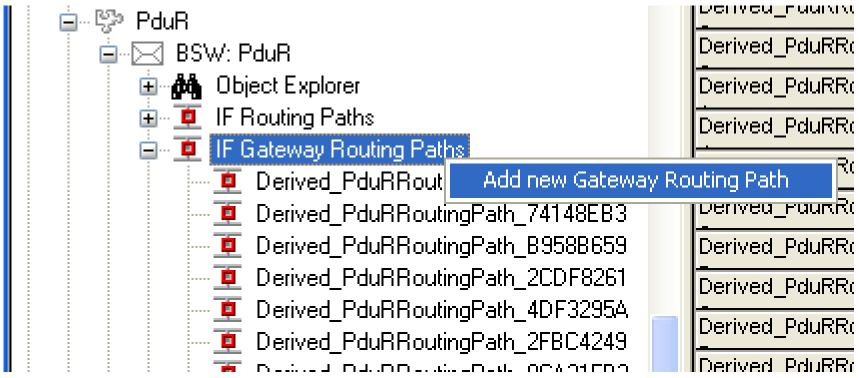

PduR
BSW: PduR
Object Explorer
IF Routing Paths
IF Gateway Routing Paths
Add new Gateway Routing Path
Derived_PduRRout
Derived_PduRRoutingPath_74148EB3
Derived_PduRRoutingPath_B958B659
Derived_PduRRoutingPath_2CDF8261
Derived_PduRRoutingPath_4DF3295A
Derived_PduRRoutingPath_2FBC4249
Derived_PduRRc

Figure 6-6 Creation of an interface layer routing path

A blank routing path will be added to the “Interface Layer Routing Relations” or “TP Layer Routing Relations” list where further settings have to be made.

Info

1:N routing paths are configured by N independent 1:1 routing paths using the same source I-PDU.

Interface Routing: Please note that all properties of N Interface routing relations must be equal – except the source I-PDU reference.

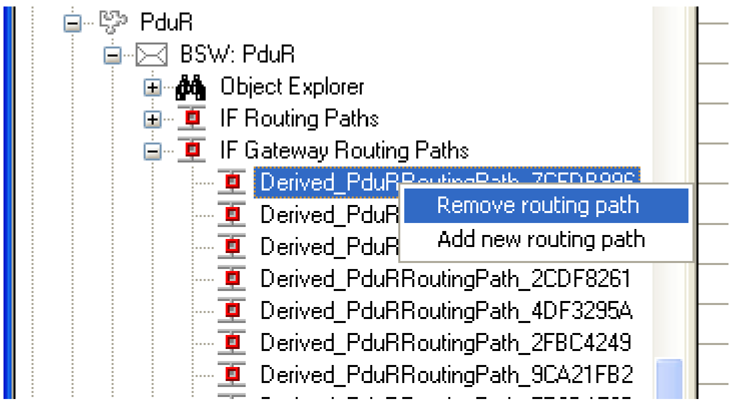
Function Requests: It is also possible to forward a functional request to the local DCM and to route the same PDU to one or several destination PDUs using the TP gateway.

Physical Requests: must always be routed in a 1:1 manner and can either be routed to another TP I-PDU or can be forwarded to DCM.

N:1 routing paths where several source I-PDUs are mapped to one destination I-PDU are not supported.

6.2.3.2 Removing a routing path

In order to delete an IF or TP routing path, right click on the routing path and select “Remove routing path”. The routing path will be deleted permanently.

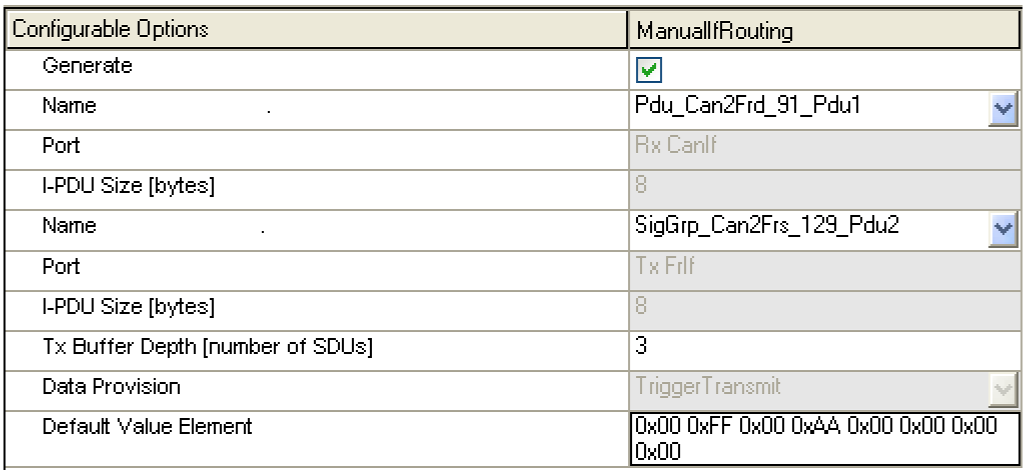


PduR
BSW: PduR
Object Explorer
IF Routing Paths
IF Gateway Routing Paths
Derived_PduRRoutingPath_7CFDB995
Derived_PduR
Derived_PduR
Derived_PduRRoutingPath_2CDF8261
Derived_PduRRoutingPath_4DF3295A
Derived_PduRRoutingPath_2FBC4249
Derived_PduRRoutingPath_9CA21FB2

Figure 6-7 Removing a Routing Path

6.2.3.3 Interface Gateway Routing Paths

An IF gateway routing path requires the following configuration settings. Depending on the type of routing path, not all attributes are configurable.



Configurable Options	ManualIfRouting
Generate	✓
Name	Pdu_Can2Frd_91_Pdu1
Port	Rx CanIf
I-PDU Size [bytes]	8
Name	SigGrp_Can2Frs_129_Pdu2
Port	Tx FrIf
I-PDU Size [bytes]	8
Tx Buffer Depth [number of SDUs]	3
Data Provision	TriggerTransmit
Default Value Element	0x00 0xFF 0x00 0xAA 0x00 0x00 0x00 0x00

Figure 6-8 IF Gateway Routing Path Parameters

Attribute Name	Value Type	Values The default value is written in bold	Description
Generate	Boolean	**True** False	> True: the routing path will be generated > False: The generator will ignore the routing path. This option can be used to temporarily deactivate a routing path without deleting it.
Name	Object		Select the Source I-PDU of the routing path. The drop down list displays all IF I-PDUs that are assigned to the PduR or to no module at all. I-PDUs that are handled by other module such as XCP or TP cannot be routed by the interface layer gateway and are not included in the drop down list.
Port	String		Visualizes the port of the source I-PDU.

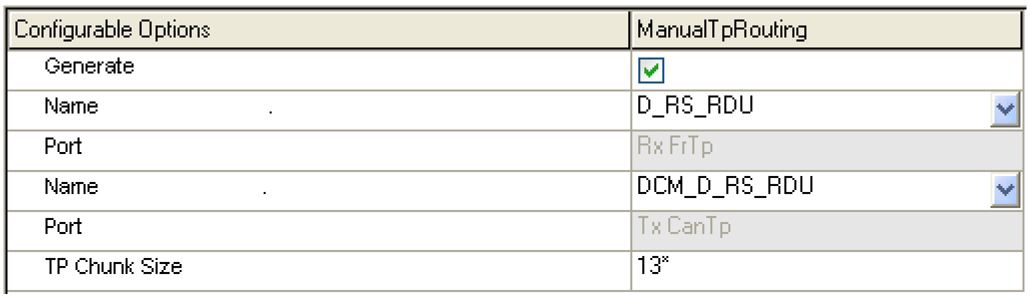
Attribute Name	Value Type	Values The default value is written in bold	Description
I-PDU Size [bytes]	Integer		The size indicates the payload length of the source I-PDU. The gateway always routes the greatest common number of bytes from the source I-PDU to the destination I-PDU.
Name	Object		Select the Destination I-PDU of the routing path. The drop down list displays all IF I-PDUs that are assigned to the PduR or to no module at all. I-PDUs that are handled by other module such as NM, XCP or TP cannot be routed by the interface layer gateway and are not included in the drop down list.
Port	String		Visualizes the port of the destination I-PDU.
I-PDU Size [bytes]	Integer		The size indicates the payload length of the destination I-PDU. The gateway always routes the greatest common number of bytes from the source I-PDU to the destination I-PDU.
Callout Function	String		Configure a callout function that is invoked before the IF routing is performed. It is possible to configure one callout function for several routing paths. If no callout function has been configured, the routing will be executed always. In case of a 1:N routing a callout function can be configured independently for each of the N routing paths. During routing, the callout functions will be called for any of the N routing paths independently of the returned values.
Tx Buffer Depth [number of SDUs]	Integer		This option allows specifying the depth of the PDU Router queue. A depth of 0 or 1 defines a routing without the usage of a FiFo queue. This queuing strategy will always transmit the latest data and will overwrite data that has not been transmitted if a new instance has been received. A depth greater than 2 defines a FiFo queued routing with flush-on-overrun behavior. I.e. in case of a buffer overrun, the buffer will be flushed before the latest I-PDU is added to the queue. The maximum queue depth is limited to 255 I-PDUs for each Tx queue.
Data Provision	Enum		The type of data provision is chosen automatically depending on the destination I-PDU: > Direct: Direct data provision is chosen for all CAN IF I-PDUs and FlexRay IF PDUs that use the immediate transmission. Each queued I-PDU is

Attribute Name	Value Type	Values The default value is written in bold	Description
			transmitted once until the queue is empty or an error occurs. If the queue is empty no further transmission requests are issued. On a CAN bus this results in the I-PDU no longer being transmitted. On a FlexRay bus, the I-PDU update bit remains cleared until new data is queued. > TriggerTransmit: Trigger Transmit data provision is configured for all LIN, FlexRay and MOST IF I-PDUs that do no use the immediate transmission. Each queued I-PDU is transmitted once until the queue is empty or an error occurs. If the queue is empty no further transmission requests are issued. On a LIN bus, this results in the last I-PDU instance being transmitted in each frame slot until the schedule table is changed or new data is received. On a FlexRay bus, the I-PDU update bit remains cleared until new data is queued.
Default Value	String		The default value is only relevant to the routing paths that use the trigger-transmit data provision. The value is used by PDU Router if the interface layer (i.e. LIN IF, FR IF, MOST IF) requests an SDU before the source I-PDU was received for the first time. If possible the value is retrieved automatically (e.g. from LIN .ldf files). If the default value is configured manually, the following format has to be used: > Each byte is configured separately using hexadecimal values (e.g. 0x1A) > The bytes are separated by a space symbol > For each SDU byte one default value element is required Example: 0x12 0x23 0xAB 0xFF

Table 6-2 IF Gateway Routing Path Parameters

6.2.3.4 Transport Protocol Gateway Routing Paths

A TP gateway routing path requires the following configuration settings. Depending on the type of routing path, not all attributes are configurable.



Configurable Options	ManualTpRouting
Generate	✓
Name	D_RS_RDU
Port	Rx FrTp
Name	DCM_D_RS_RDU
Port	Tx CanTp
TP Chunk Size	13*

Figure 6-9 TP Gateway Routing Path Parameters

Attribute Name	Value Type	Values The default value is written in bold	Description
Generate	Boolean	**True** False	> True: the routing path will be generated > False: The generator will ignore the routing path. This option can be used to temporarily deactivate a routing path without deleting it.
Name	Object		Select the Source I-PDU of the routing path. The drop down list displays all TP layer I-PDUs that are available for routing. The TP SDUs have to be configured in the TP. I-PDUs of addressing type “Physical” (physical requests) that are received by an Upper Layer must not be routed.
Port	String		Visualizes the port of the source I-PDU.
Name	Object		Select the Destination I-PDU of the routing path. The drop down list displays all TP layer I-PDUs that are available for routing. The TP SDUs have to be configured in the TP. Destination I-PDUs that are transmitted by an Upper Layer must not be routed.
Port	String		Visualizes the port of the destination I-PDU.
TP Chunk Size	Integer		The TP chunk size determines the number of bytes that must be received before the transmission of the destination I-PDU is triggered. Thus, this parameter can be used to determine whether an I-PDU shall be routed “on-the-fly” (chunk size < I-PDU length) or “store-and-forward” (chunk size >= I-PDU length). This Parameter has to be configured on each routing relation. 1:N routing paths require an equal “TP Chunk Size” for all N routing paths.

Table 6-3 TP Gateway Routing Path Parameters

Caution

Routing relations of functional requests (1:1 and 1:N) require a “TP Chunk Size” of at least the size of the largest request that will to be routed by this routing path.

If this is not given, routing and forwarding of the Rx I-PDU to DCM will fail.

In case of 1:N routing paths, the “TP Chunk Size” of all N routing paths with the same source I-PDU must have the same size.

6.2.4 TP Buffers

The TP gateway requires at least one buffer element to allow the routing of TP I-PDUs. The size of the largest TP SDU that can be routed by the TP gateway is restricted by the size of the largest TP buffer element configured.

The number buffer elements that are to be configured depend on the number of TP I-PDUs that shall be routable at the same time. For each 1:N routing path only one buffer element has to be encountered. As the buffer elements are not connected with a routing path, the number of configured buffer elements can be smaller than the number of routing paths.

The size of each buffer element is configured individually and therefore allows fine adaptations according to the use case. The sizes of the configured buffer depend, first of all, on the chunk sizes of the configured routing paths. When the gateway assigns a buffer element to a routing path it is required that the assigned buffer has at least the size of the TP chunk size.

Increase the PduR buffer size to adapt a high bus load on the reception side to a lower bandwidth on the transmit side.

Since there is no connection between a routing path and a buffer element at configuration time, the gateway can allocate the buffer elements dynamically at runtime.

Thereby the gateway uses the following rules to choose one of the configured TP buffers.

If the size of the incoming I-PDU is smaller than the configured TP chunk size the smallest available buffer is used that can hold the complete I-PDU.

Otherwise, the chosen buffer has at least the size of the TP chunk.

Among the potential buffers the gateway tries to find a buffer with at least the size of the I-PDU that is to be routed. If such a buffer could not be found the next smaller unused buffer is assigned to the routing path.

If no buffer can be found the buffer request is rejected and no routing of the received TP I-PDU takes place.

To allow parallel reception and transmission of a TP I-PDU, it is recommended that buffers are provided with a size of at least twice the size of the TP chunk size. The doubled size allows the gateway to first fill one part of the buffer, than start the transmission of the received data while using the second buffer part for further reception.

Due to the ring-buffer mechanism, it is not required that a buffer element with the size of the largest expected TP I-PDU is configured. A buffer smaller than the routed I-PDU can, however, result in wait-frames or a buffer-overflow if the destination connection is slower that the source connection. A buffer overflow can be avoided if the receiving TP connection allows dynamic adaptation of the block size (BS) (e.g. CanTp connection with BS greater 0). If a BS of 0 is used by some of the source TP connections, the configured buffer sizes should be dimensioned in a way that buffer overflows are avoided.

To avoid allocation of a large and thus expensive buffer to a small TP I-PDU (e.g. single frame I-PDUs), it is advisable to configure at least one small buffer with the size of a single frame payload (e.g. 7 bytes) due to a buffer is locked until the routing is finished.

To provide fast buffer for the reception Tp, the PduR could route systematically each single buffer-chunk that has been received, if “PdurBalanceRoutingTime” is configured. This is done independently from the amount of buffer chucks that have been actually received. This optimization is important if the transmit side is slower than the receiver side.

6.2.4.1 Creating a TP Buffer

To create a TP Buffer, right click on the “TP Buffers” tree in the GENy module tree and select “Add new TP Buffer”.

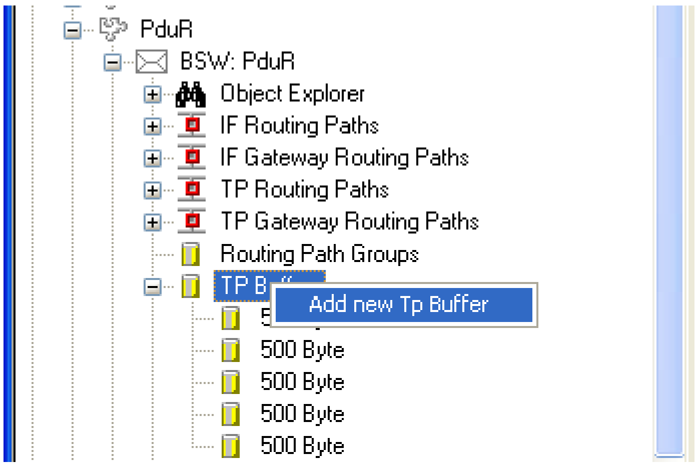


Figure 6-10 Creating a TP Buffer

6.2.4.2 Removing a TP Buffer

To remove a TP Buffer, right click on the TP Buffer and select “Remove buffer element”. The buffer element will be deleted permanently.

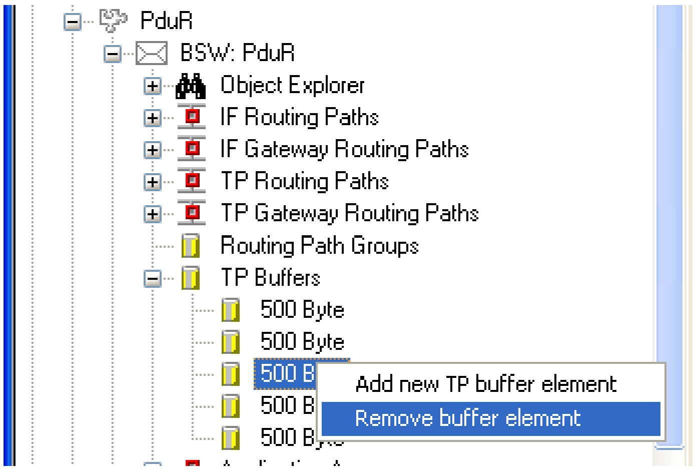


Figure 6-11 Remove a TP Buffer

6.2.4.3 TP Buffer Properties

A blank buffer element will be added to the TP Buffer list. Initially the buffer has a length of 0 bytes which is an invalid value. The generator ignores TP buffer elements with a size of 0. To configure the size of a TP buffer element select the element from the TP buffer tree and key in the required buffer size in bytes.

Attribute Name	Value Type	Values The default value is written in bold	Description
Length [byte]	Integer		Configure the size in bytes number of the TP Buffer. If the feature to fulfill ISO timing requirements is activated in your delivery PDUR_<LoTp>PROVIDERXBUFFER_CALLS is generated in PduR_Cfg.h to STD_ON the assigned buffer shall have at least the size of the TP chunk size + First Frame length of the TPs.

Table 6-4 TP Buffer Properties

6.2.5 IF and TP Application Access

This PDU Router feature can be used to access lower layer I-PDUs directly from above the PDU Router. The feature is an optional extension to the AUTOSAR standard.

The GENy GUI allows assigning lower layer PDUs to the application by choosing the appropriate PDU and assigning the handled ID. The configured handle ID is used to identify a PDU through the application interface. The application can, for compatibility reasons, specify the handle ID for Rx and Tx application I-PDUs.

6.2.5.1 Creating an Application I-PDU

Four different types of application access PDUs can be configured: Rx and Tx interface layer I-PDUs as well as Rx and Tx TP layer I-PDUs. Each type represented in an own tree element in the GENy GUI.

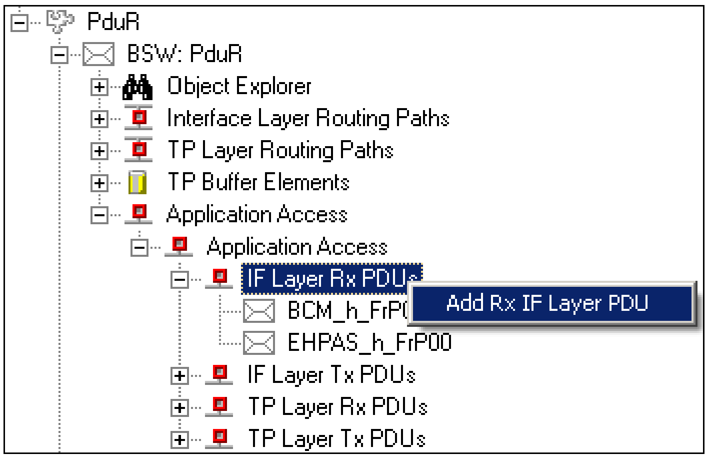


Figure 6-12 Creating an Application Access I-PDU

An application access PDU is created by right clicking on the appropriate tree element and choosing “Add [Rx|Tx] [IF|TP] layer PDU”. A blank (invalid) element will be added to the tree.

6.2.5.2 Removing an Application I-PDU

Application access PDUs can be removed by right clicking on the appropriate element and choosing “Remove [Rx|Tx] [IF|TP] layer PDU”.

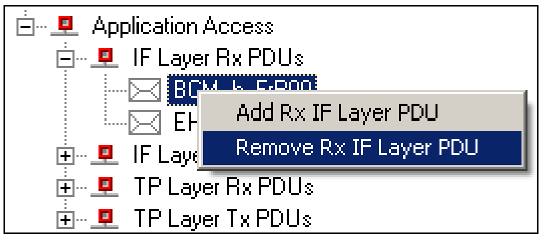


Figure 6-13 Removing an Application Access I-PDU

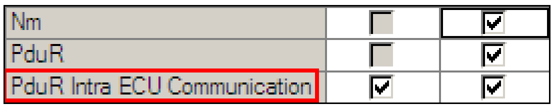
6.2.5.3 Application I-PDU Properties

Attribute Name	Value Type	Values The default value is written in bold	Description
Name	String		Select the PDU that shall be accessible for the application. The chosen PDU must not be used by any other higher layer such as COM or DCM.
Handle Id	Integer		The handle Id can be determined by the application. The handle space for any type of application access PDUs must be zero based and continuous.

Table 6-5 Application Access I-PDU Properties

6.3 Intra ECU Communication

To enable intra ECU communication, the GENy module “PduR Intra ECU Communication” has to be enabled in the GENy modules list first. In general this module is enabled automatically when enabling the “PduR” module. Please note that the module “PduR Intra ECU communication has no channel dependency and is therefore enabled or disabled for all channels at the same time.



Nm	☐	✓
PduR	☐	✓
PduR Intra ECU Communication	✓	✓

Figure 6-14 Enabling intra ECU communication in GENy

This GENy module allows creating TP I-PDU objects that can be used for intra ECU communication. Semantically these objects are identical to e.g. CanTp or FrTp I-PDUs. Once created the intra ECU I-PDUs can be referred from upper layer modules such as DCM or ApplTp.

Two different types of intra ECU I-PDUs can be created:

> Request I-PDUs: Can be configured as DCM Rx I-PDUs and ApplTp Tx I-PDUs.

> Response I-PDUs: Can be configured as ApplTp Rx I-PDUs and DCM Tx I-PDUs.

As the handling of request and response intra ECU communication TP I-PDUs is similar, only the handling of request I-PDUs is depicted here.

6.3.1.1 Creating Intra ECU communication TP I-PDUs

Before a communication relation can be configured in the ApplTp interface or the DCM module, it is required to create an internal PDU.

To create an internal PDU open the GENy module “PduR Intra ECU Communication” and right mouse click on the related PDU collection. Choose “Add new internal TP I-PDU”.

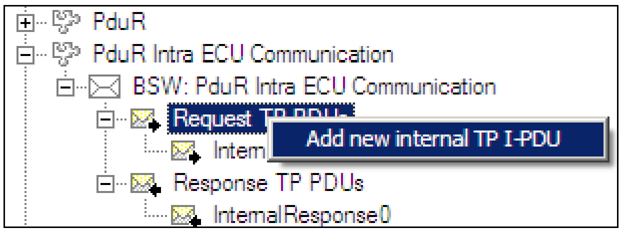

PduR
PduR Intra ECU Communication
BSW: PduR Intra ECU Communication
Request TP PDUs
Intern
Add new internal TP I-PDU
Response TP PDUs
InternalResponse0

Figure 6-15 Creating Intra ECU communication TP I-PDUs

A new PDU is created that can thereafter be selected referred from DCM connections or the ApplTp interface. The name of an internal PDU can be modified by the user if required.

It is up to the user to ensure that the name of a created PDU is unique within the ECU configuration.

6.3.1.2 Removing Intra ECU communication TP I-PDUs

To remove an I-PDU used for internal communication, right click on the PDU to be deleted and choose “Remove internal TP I-PDU”. The PDU will be deleted permanently and removed from all entities that referred to it.

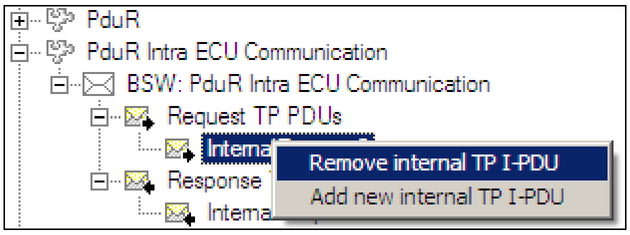

PduR
PduR Intra ECU Communication
BSW: PduR Intra ECU Communication
Request TP PDUs
Interna
Response
Interna
Remove internal TP I-PDU
Add new internal TP I-PDU

Figure 6-16 Removing Intra ECU communication TP I-PDUs

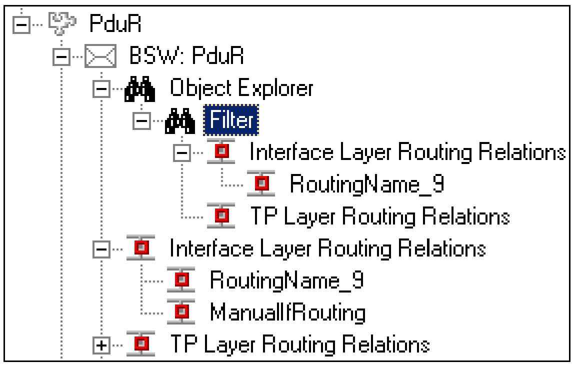
6.3.1.3 Intra ECU communication Properties

Attribute Name	Value Type	Values The default value is written in bold	Description
Maximum TP Connection Number	Integer	1	Defines the maximum number of TP I-PDUs that can be transferred concurrently using intra ECU communication. The feature is relevant for allowing two upper layer components (i.e. ApplTp and DCM) to communicate with each other.
Configuration Time Base [sec]	Float	0.001	Configure the cycle time in which PduR_MainFunction() has to be called.
Rx Buffer Response Timeout [sec]	Float	0.2	Configure maximum time of consecutive ProvideRxBuffer calls with the return value BUFREQ_E_BUSY. If the timeout timer is expired, the Intra ECU communication connection will be terminated.
Tx Buffer Response Timeout [sec]	Float	0.2	Configure maximum time of consecutive ProvideTxBuffer calls with the return value BUFREQ_E_BUSY. If the timeout timer is expired, the Intra ECU communication connection will be terminated.

Table 6-6 Intra ECU communication Properties

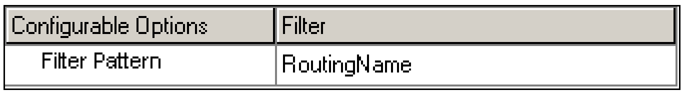
6.4 Object Explorer

The “Object Explorer” tree can be used to filter Gateway Routing Paths based on names and special filter commands. The filter can be used for IF and TP Gateway Routing Paths.



```
PduR
  BSW: PduR
    Object Explorer
      Filter
        Interface Layer Routing Relations
          RoutingName_9
        TP Layer Routing Relations
    Interface Layer Routing Relations
      RoutingName_9
      ManualIfRouting
    TP Layer Routing Relations
```

Figure 6-17 Object Explorer in GENy



Configurable Options	Filter
Filter Pattern	RoutingName

Figure 6-18 Object Explorer Properties

Attribute Name	Value Type	Values The default value is written in bold	Description
Filter Pattern	String		> Name based search: The object explorer lists all routing paths that contain the filter pattern string. Thereby the routing path name and the source and destination PDU name are evaluated. > Special search commands: The filter provides some special search commands that can be used to search for a dedicated class of routing paths: > %auto: shows all automatic routing paths (i.e. routing paths that have been detected by OEM specific rules). > %man: shows all manual routing paths (i.e. routing paths imported from an EcuC file or created by the user manually). > %disabled: filters all routing paths which cannot be generated (generate flag is disabled).

Table 6-7 Object Explorer Properties

6.5 Configuration in EcuC Data Base

The following chapters contain the detailed description of the EcuC configuration parameters for the PduR.

6.5.1 PduR

Container Name	PduR
Path	\MICROSAR\PduR
Multiplicity	0...1
Description	Configuration of the PduR (PDU Router) module.

Table 6-8 Container PduR

6.5.1.1 PduRBswModules

Container Name	PduRBswModules
Path	\MICROSAR\PduR\PduRBswModules
Multiplicity	0...*
Description	Each container describes a specific BSW module (upper/CDD/lower/IpduM) that the PDU Router shall interface to.

Table 6-9 Container PduRBswModules

Attribute Name	**Multi-plicity**	**Value Type**	**Values** Default value is typed bold	**Description**
PduRCommunicationInterface	1...1	Boolean	true false	Specifies if the BSW module supports the Communication Interface APIs or not. Value true the APIs are supported.
PduRLowerModule	1...1	Boolean	true false	The PduRLowerModule will decide who will call the APIs and who will implement the APIs.
PduRTransportProtocol	1...1	Boolean	true false	The PDU Router module shall use the API parameters specified for transport protocol interface.
PduRTriggertransmit	1...1	Boolean	true false	Specifies if the BSW module supports the TriggerTransmit API or not. Value true the API is supported.
PduRUpperModule	1...1	Boolean	true	The PduRUpperModule will decide who will call

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
			false	the APIs and who will implement the APIs.
PdurCancelReceive	1...1	Boolean	true false	Specifies if the Transport protocol module supports the CancelReceive API or not. Value true the API is supported.
PduRCancelTransmit	1...1	Boolean	true false	Specifies if the BSW module supports the CancelTransmit API or not. Value true the API is supported.
PduRChangeParameterRequestApi	1...1	Boolean	true false	Specifies if the BSW module supports the CancelTransmit API or not. Value true the API is supported.

Table 6-10 Attributes of PduRBswModules

6.5.1.2 PduRGeneral

Container Name	PduRGeneral
Path	\MICROSAR\PduR\PduRGeneral
Multiplicity	1...1
Description	This container is a subcontainer of PduR and specifies the general configuration parameters of the PDU Router.

Table 6-11 Container PduRGeneral

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRCanIfSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BSW module CanIf. This parameter must be preconfigured.
PduRCanTpSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BSW module CanTp. This parameter must be preconfigured.
PduRMostTpSupport	1...1	Boolean	true false	Enable / Disable PduR support for the module MostTp. This parameter must be preconfigured. This feature is an extension to the PduR module.
PduRMostIfSupport	1...1	Boolean	true false	Enable / Disable PduR support for the module MostIf. This parameter must be preconfigured. This feature is an extension to the PduR module.
PduRComSupport	1...1	Boolean	true	Enable / Disable PduR support for the BSW

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
			false	module Com. This parameter must be preconfigured.
PduRDcmSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BMW module Dcm. This parameter must be preconfigured.
PduRDevErrorDetect	1...1	Boolean	true false	Enable / Disable BSW module Development Error Detection and Notification.
PduRFifoTxBufferSupport	1...1	Boolean	true false	Enable / Disable PduR support for FIFOs transmit buffers; required only for gateway operations.
PduRFrIfSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BSW module FlexrayIf. This parameter must be preconfigured.
PduRFrTpSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BSW module FlexrayTp. This parameter must be preconfigured.
PduRGatewayOperation	1...1	Boolean	true false	Enable / Disable PduR support for gateway operations. If, Tp gateways support inclusive. The .dlls, the Gw_AsrPduRGw.dll (If) and Gw_AsrPduRGwTp.dll (Tp) must be available. This switch allows disabling the gateway feature for configurations that include the gateway.
PduRIpduMSupport	1...1	Boolean	true false	Enable / Disable PduR support for the Lower layer interface of the BSW module IPdu Multiplexer. This parameter must be preconfigured.
PduRIpduMUpSupport	0...1	Boolean	true false	Enable / Disable PduR support for the upper layer interface of the BSW module IPdu Multiplexer. This parameter must be preconfigured.
PduRLinIfSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BSW module LinIf. This parameter must be preconfigured.
PduRLinTpSupport	1...1	Boolean	true false	Enable / Disable PduR support for the BSW module LinTp. This parameter must be preconfigured.
PduRMemorySize	1...1	Integer	**0** - n	Reserves buffer. Should post build configurations require additional buffer, it should be allocated at linktime. Parameter required for gateway operation only.
PduRMulticastFromIfSupport	1...1	Boolean	true false	Configuration parameter to enable or disable PDU Router support for multicasts from an interface module to upper layer modules or lower layer interface modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRMulticastFromTpSupport	1...1	Boolean	true false	Configuration parameter to enable or disable PDU Router support for multicasts from a TP module to upper layer modules or lower layer TP modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.
PduRMulticastToIfSupport	1...1	Boolean	true **false**	This Parameter is not used by the PduR currently. Configuration parameter to enable or disable PDU Router support for multicasts from an upper layer module to interface modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.
PduRMulticastToTpSupport	1...1	Boolean	true **false**	This Parameter is not used by the PduR currently. Configuration parameter to enable or disable PDU Router support for multicasts from an upper layer module to TP modules; if PDUR_ZERO_COST_OPERATION is enabled, this parameter has to be disabled.
PduRSbTxBufferSupport	1...1	Boolean	true false	This Parameter is not used by the PduR currently. Configuration parameter to enable or disable PDU Router support for single buffers as PDU transmit buffers; if PDUR_GATEWAY_OPERATION is disabled, this parameter has to be disabled.
PduRVersionInfoApi	1...1	Boolean	true false	Activates/Deactivates the Version Info API.
PduRZeroCostOperation	1...1	Boolean	true **false**	This Parameter is not used by the PduR currently. All routing paths are implicitly defined and the communication modules directly above or below the PDU Router shall directly call each other without using PDU Router functions (zero cost operation). The configuration parameters PDUR_SINGLE_IF and PDUR_SINGLE_TP are used to specify the related lower layer module.
PduRPBStartAddress	0...1	Integer	not specified	This is the start address of the memory block in ROM where the post build configuration data is located. Required for variant post-build only.
PduRProdErrorDetect	0...1	Boolean	**true** false	Enable to report production relevant errors to the Diagnostics Event Manager (DEM). If disabled no errors will be reported to DEM.
PduRRouting2CSFSupport	0...1	Boolean	true **false**	Enable / Disable support for the BSW module CanTp 4.0 API. Accepts a new routing request for a single frame while the previous is being

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
				routed.
PduRUserConfigurationFile	0...1	String	not specified	Choose a user configuration file whose content will be appended to the generated PduR_Cfg.h at the end overwritting configuration settings in PduR_Cfg.h.
PduRMaxNumberOfRxCanTpPdus	0...1	Integer	1 - 65535	Define the maximum number of CanTp Routings possible.
PduRMaxNumberOfRxFrTpPdus	0...1	Integer	1 - 65535	Define the maximum number of FrTp Routings possible.
PduRJ1939Support	0...1	Boolean	true **false**	Enable / Disable PduR support for the module J1939Tp. This parameter must be preconfigured. This feature is an extension to the PduR module.
PduRApplIfSupport	0...1	Boolean	true **false**	Enable / Disable PduR support for direct application access to If layer PDUs in the PduR. This parameter must be preconfigured. This feature is an extension to the PduR module.
PduRApplTpSupport	0...1	Boolean	true **false**	Enable / Disable PduR support for direct application access to Tp layer PDUs in the PduR. This parameter must be preconfigured. This feature is an extension to the PduR module.
PduRSoAdTpSupport	0...1	Boolean	true **false**	Enable / Disable PduR support for module SoAd interface access to Tp layer PDUs. This feature is an extension to the PduR module.
PduRApplTpReceiveCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the Application's cancel Tp receive request feature. This feature is an extension to the PduR module.
PduRDcmReceiveCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module Dcm's cancel receive request feature. This feature is an extension to the PduR 3.0 module.
PduRFrTpReceiveCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module FrTp's cancel receive request feature. This feature is an extension to the PduR 3.0 module.
PduRCanTpReceiveCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module CanTp's cancel receive request feature. This feature is an extension to the PduR 3.0 module.
PduRLinTpReceiveCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module LinTp's cancel receive request feature. This feature is an extension to the PduR 3.0 module.
PduRApplTpTransmitCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the Application's cancel Tp transmit request feature. This feature is an extension to the

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
				PduR module.
PduRDcmTransmitCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module Dcm's cancel transmit request feature. This feature is an extension to the PduR module.
PduRFrTpTransmitCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module FrTp's cancel transmit request feature. This feature is an extension to the PduR module.
PduRCanTpTransmitCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module CanTp's cancel transmit request feature. This feature is an extension to the PduR module.
PduRLinTpTransmitCancellation	0...1	Boolean	true **false**	Enable / Disable PduR support for the BSW module LinTp's cancel transmit request feature. This feature is an extension to the PduR module.
PduRCanNmSupport	1...1	Boolean	true false	Enable / Disable PduR support for module CanNm interface access to If layer PDUs. This feature is an extension to the PduR module.
PduRNmOsekSupport	1...1	Boolean	true false	Enable / Disable PduR support for module NmOsek interface access to If layer PDUs. This feature is an extension to the PduR module.
PduRFrNmSupport	1...1	Boolean	true false	Enable / Disable PduR support for module FrNm interface access to If layer PDUs. This feature is an extension to the PduR module.
PduRMaxRoutingPathGroups	0...1	Integer	0 - 65535	Defines the maximum number of PduRRoutingPathGroups in one Identity.
PduRMaxRoutingPathsInRPGs	0...1	Integer	0 - 65535	Defines the maximum number of PduRRoutingPaths in PduRRoutingPathGroups in one Identity.
PduRRoutingPathGroupSupport	0...1	Boolean	true **false**	Enable / Disable PduR support for PduRRoutingPathGroups.
PduRFrTp2ProvideRxBufferCalls	0...1	Boolean	true **false**	Enable / Disable PduR support for CanTp to support two ProvideRxBuffer calls
PduRCanTp2ProvideRxBufferCalls	0...1	Boolean	true **false**	Enable / Disable PduR support for CanTp to support two ProvideRxBuffer calls.
PduRUsePduInfoType	0...1	Boolean	true **false**	If this feature is activated, PduInfoPtr instead of SduPtr is used for the APIs PduR_<Lo>RxIndication, PduR_<Lo>TriggerTransmit, <Up>_RxIndication and <Up>_TriggerTransmit.
PduRBalanceRoutingTime	0...1	Boolean	true **false**	Multiple Chunks of the TP buffer are blocked by the Tx TP until the data is transmitted. The Rx TP has to wait until the buffer is free. The TP timings have to take this effect into account and

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
				the timings have to be very tolerant and the routing latency is reduced. Due to this, the TP transmission shall be managed by the PDUR with chunks to provide fast buffer for the reception TP.
PduRDynamicDlcSupport	0...1	Boolean	true **false**	If this feature is activated, the received data length (PduInfoPtr->SduLength) is checked in the function PduR_<Lo>RxIndication and only completely received bytes are processed.

Table 6-12 Attributes of PduRGeneral

6.5.1.3 PduRGlobalConfig

Container Name	PduRGlobalConfig
Path	\MICROSAR\PduR\PduRGlobalConfig
Multiplicity	1...1
Description	This container contains the global configuration parameter of the PduR. It is a MultipleConfigurationContainer, i.e. this container and its sub-containers exit once per configuration set.

Table 6-13 Container PduRGlobalConfig

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRConfigurationId	0...1	Integer	not specified	Unique post-build configuration identifier.

Table 6-14 Attributes of PduRGlobalConfig

Sub Container	Multiplicity
PduRRoutingTable	1...1
PduRTpBufferTable	0...1
PduRTxBufferTable	0...1
PduRApplicationAccess	0...1

Table 6-15 Sub Containers of PduRGlobalConfig

6.5.1.4 PduRRoutingTable

Container Name	PduRRoutingTable
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable
Multiplicity	1...1
Description	PDU Router routing table is a subcontainer ofPduR. This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_ZERO_COST_OPERATION is disabled.

Table 6-16 Container PduRRoutingTable

Sub Container	Multiplicity
PduRRoutingPath	0...*
PduRRoutingPathGroup	0...*

Table 6-17 Sub Containers of PduRRoutingTable

6.5.1.5 PduRRoutingPath

Container Name	PduRRoutingPath
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable\PduRRoutingPath
Multiplicity	0...*
Description	This container is a subcontainer of PduRRoutingTable and specifies the routing path of a PDU.

Table 6-18 Container PduRRoutingPath

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
SduLength	0...1	Integer	0 - 255	This Parameter is not used. Length of PDU data (SDU). Only required if a Tx buffer is configured.
TpChunkSize	0...1	Integer	1... **7** ...65535	Chunk size for routing on the fly. Defines the number of bytes which shall be received before transmission on the destination bus may start. Only required for TP gateway PDUs. The TpChunkSize shall not be larger than the length of the related TP Buffer.

Table 6-19 Attributes of PduRRoutingPath

Sub Container	Multiplicity
PduRDefaultValue	0...1
PduRDestPdu	1...*
PduRSrcPdu	1...1

Table 6-20 Sub Containers of PduRRoutingPath

6.5.1.6 PduRDefaultValue

Container Name	PduRDefaultValue
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable\PduRRoutingPath \PduRDefaultValue
Multiplicity	0...1
Description	This container is a subcontainer of PduRRoutingPath and specifies the default value of the I-PDU. Only required for gateway operation and if at least one PDU specified by PduRDestPdu uses TriggerTransmit Data provision. Represented as an array of IntegerParamDef.

Table 6-21 Container PduRDefaultValue

Sub Container	Multiplicity
PduRDefaultValueElement	0...*

Table 6-22 Sub Containers of PduRDefaultValue

6.5.1.7 PduRDefaultValueElement

Container Name	PduRDefaultValueElement
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable\PduRRoutingPath \PduRDefaultValue\PduRDefaultValueElement
Multiplicity	0...*
Description	Each value element is represented by the element and the position in an array.

Table 6-23 Container PduRDefaultValueElement

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
DefaultValueElement	1...1	Integer	0 - 255	The default value consists of a number of elements. Each element is one byte long and the number of elements is specified by SduLength. The position of this parameter in the container is specified by the ElementBytePosition parameter.
ElementBytePosition	1...1	Integer	not specified	This parameter specifies the byte position of the element within the default value.

Table 6-24 Attributes of PduRDefaultValueElement

6.5.1.8 PduRDestPdu

Container Name	PduRDestPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable\PduRRoutingPath \PduRDestPdu
Multiplicity	1...*
Description	This container is a subcontainer of PduRRoutingPath and specifies one destination for the PDU to be routed.

Table 6-25 Container PduRDestPdu

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
DataProvision	0...1	Enumeration	DIRECT TRIGGER_TRANSMIT	Specifies how data are provided: direct (as part of the Transmit call) or via the TriggerTransmit callback function. Only required for non-TP gateway PDUs.
RoutingCallout	0...1	Function	not specified	Configure a callout function implemented by the application and triggered on the routing event. The return value from the function determines if the routing was executed (E_OK) or canceled (E_NOT_OK). The prototype of the function is provided in PduR_Notifications.h.
DisableRouting	0...1	Boolean	true **false**	This parameter allows to suspend a routing without deleting it. If the parameter is not present or set to false, the routing will be generated and active. This parameter is evaluated for TP- and interface layer routing

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
				relations.
PduRCddDobtSwitc h	0...1	Boolean	true **false**	True, if the Tx CanTp Port can be used by CddDobt.
DestPduRef	1...1	Reference	not specified	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.
TxBufferRef	0...1	Reference	not specified	Specifies the assigned transmit buffer. Only required for specific non-TP gateway PDUs.

Table 6-26 Attributes of PduRDestPdu

6.5.1.9 PduRSrcPdu

Container Name	PduRSrcPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable\PduRRoutingPath \PduRSrcPdu
Multiplicity	1...1
Description	This container is a subcontainer of PduRRoutingPath and specifies the source of the PDU to be routed.

Table 6-27 Container PduRSrcPdu

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
HandleId	1...1	Integer	0 - 65535	PDU identifier assigned by PDU Router.
PduRCddDobtSwitc h	0...1	Boolean	true **false**	True, if the Rx CanTp Port can be used by CddDobt.
SrcPduRef	1...1	Reference	not specified	Source PDU reference; reference to unique PDU identifier which shall be used for the requested PDU Router operation.

Table 6-28 Attributes of PduRSrcPdu

6.5.1.10 PduRRoutingPathGroup

Container Name	PduRRoutingPathGroup
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRRoutingTable\PduRRoutingPath Group
Multiplicity	0...*
Description	This container groups routing path destinations. Destinations areused instead of routing paths since a routing path can be 1:n. It isdesirable to be able to enable/disable a specific bus (i.e. adestination) rather than a routing path. Of course it is possible tocreate groups that covers specific routing paths as well. Enabling anddisabling of routing path groups are made using the PduR API

Table 6-29 Container PduRRoutingPathGroup

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRRoutingPathGroupId	1...1	Integer	0 - 65535	Identification of the routing group. The identification will be used by the PduR_EnableRouting() / PduR_DisableRouting() API in the PDU Router module API. GENy: The numerical value used as the ID of this Pdu Group. The PduRRoutingPathGroupId is required by the PduR
PduRIsEnabledAtInit	1...1	Boolean	true **false**	If set to true this routing path group will be enabled after initialization of the PDU Router module.
PduRRoutingPathRef	1...*	Reference	not specified	This container is a subcontainer of PduRRoutingPath and specifies the default value of the I-PDU. Only required for gateway operation and if at least one PDU specified by PduRDestPdu uses TriggerTransmit Data provision. Represented as an array of IntegerParamDef.

Table 6-30 Attributes of PduRRoutingPathGroup

6.5.1.11 PduRTpBufferTable

Container Name	PduRTpBufferTable
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRTpBufferTable
Multiplicity	0...1
Description	This container is a subcontainer of PduR and contains the definition of all TP buffers (only required for PDU Router gateway operation). This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_GATEWAY_OPERATION is enabled.

Table 6-31 Container PduRTpBufferTable

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRMaxTpBufferNumber	1...1	Integer	0 - 65535	Maximum number of TP buffers.

Table 6-32 Attributes of PduRTpBufferTable

Sub Container	Multiplicity
PduRTpBuffer	0...*

Table 6-33 Sub Containers of PduRTpBufferTable

6.5.1.12 PduRTpBuffer

Container Name	PduRTpBuffer
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRTpBufferTable\PduRTpBuffer
Multiplicity	0...*
Description	This container is a subcontainer of PduRTpBufferTable and specifies a TP buffer.

Table 6-34 Container PduRTpBuffer

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
Length	1...1	Integer	1 - 65535	Length of the TP buffer.

Table 6-35 Attributes of PduRTpBuffer

6.5.1.13 PduRTxBufferTable

Container Name	PduRTxBufferTable
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRTxBufferTable
Multiplicity	0...1
Description	This container is a subcontainer of PduR and contains the definition of all transmit buffers (used by specific non-TP PDUs; only required for PDU Router gateway operation). This container shall only be considered by the PDU Router Configuration Generator if PduRGeneral/PDUR_GATEWAY_OPERATION is enabled.

Table 6-36 Container PduRTxBufferTable

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRMaxTxBufferNumber	1...1	Integer	0 - 65535	Maximum number of transmit buffers.

Table 6-37 Attributes of PduRTxBufferTable

Sub Container	Multiplicity
PduRTxBuffer	0...*

Table 6-38 Sub Containers of PduRTxBufferTable

6.5.1.14 PduRTxBuffer

Container Name	PduRTxBuffer
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRTxBufferTable\PduRTxBuffer
Multiplicity	0...*
Description	This container is a subcontainer of PduRTxBufferTable and specifies a transmit buffer for a non-TP PDU.

Table 6-39 Container PduRTxBuffer

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
Depth	1...1	Integer	0 - 255	Specifies the depth of the buffer
Length	1...1	Integer	1 - 255	This Parameter is not used. Length of the buffer.

Table 6-40 Attributes of PduRTxBuffer

6.5.1.15 PduRApplicationAccess

Container Name	PduRApplicationAccess
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess
Multiplicity	0...1
Description	Provides Rx and Tx PDUs that can be accessed by the application directly from the PDU Router. For any Rx and Tx PDU the reference to the global PDU and the PDU ID has to be specified. This feature is an optional Vector Informatik extension to the AUTOSAR standard. Please note that this feature can only be used if it has been licensed in the configuration tool.

Table 6-41 Container PduRApplicationAccess

Sub Container	Multiplicity
PduRApplicationPdu	0...*

Table 6-42 Sub Containers of PduRApplicationAccess

6.5.1.16 PduRApplicationPdu

Container Name	PduRApplicationPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess\PduRApplicationPdu
Multiplicity	0...*
Description	

Table 6-43 Container PduRApplicationPdu

Sub Container	Multiplicity
PduRApplicationPduType	1...1

Table 6-44 Sub Containers of PduRApplicationPdu

6.5.1.17 PduRApplicationPduType

Container Name	PduRApplicationPduType
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess\PduRApplicationPdu\PduRApplicationPduType
Multiplicity	1...1
Description	

Table 6-45 Container PduRApplicationPduType

Sub Container	Multiplicity
PduRRxIfApplicationPdu	0...1
PduRTxIfApplicationPdu	0...1
PduRRxTpApplicationPdu	0...1
PduRTxTpApplicationPdu	0...1

Table 6-46 Sub Containers of PduRApplicationPduType

6.5.1.18 PduRRxIfApplicationPdu

Container Name	PduRRxIfApplicationPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess\PduRApplicationPdu\PduRApplicationPduType\PduRRxIfApplicationPdu
Multiplicity	0...1
Description	

Table 6-47 Container PduRRxIfApplicationPdu

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRHandleId	1...1	Integer	**0** - 65535	Handle ID that is used by the application to access this PDU. The handle space must be

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
				contineous and zero based.
PduRPduRef	1...1	Reference	not specified	Reference to the global PDU. Establishes a link to the PduRRoutingPaths source interface layer I-PDU.

Table 6-48 Attributes of PduRRxIfApplicationPdu

6.5.1.19 PduRTxIfApplicationPdu

Container Name	PduRTxIfApplicationPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess\PduRApplicationPdu\PduRApplicationPduType\PduRTxIfApplicationPdu
Multiplicity	0...1
Description	The handle ID of this routing path is configured by the related PduRRoutingPath that can be found by means of the

Table 6-49 Container PduRTxIfApplicationPdu

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRHandleId	1...1	Integer	**0** - 65535	Handle ID that is used by the application to access this PDU. The handle space must be contineous and zero based.
PduRPduRef	1...1	Reference	not specified	Reference to the global PDU. Establishes a link to the PduRRoutingPaths destination interface layer PDU.

Table 6-50 Attributes of PduRTxIfApplicationPdu

6.5.1.20 PduRRxTpApplicationPdu

Container Name	PduRRxTpApplicationPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess\PduRApplicationPdu\PduRApplicationPduType\PduRRxTpApplicationPdu
Multiplicity	0...1
Description	

Table 6-51 Container PduRRxTpApplicationPdu

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRHandleId	1...1	Integer	**0** - 65535	Handle ID that is used by the application to access this PDU. The handle space must be contineous and zero based.
PduRPduRef	1...1	Reference	not specified	Reference to the global PDU. Establishes a link to the PduRRoutingPaths source TP layer PDU. Please note that TP PDUs which have been referenced here must not be involved in any other routing path.

Table 6-52 Attributes of PduRRxTpApplicationPdu

6.5.1.21 PduRTxTpApplicationPdu

Container Name	PduRTxTpApplicationPdu
Path	\MICROSAR\PduR\PduRGlobalConfig\PduRApplicationAccess\PduRApplicationPdu\PduRApplicationPduType\PduRTxTpApplicationPdu
Multiplicity	0...1
Description	

Table 6-53 Container PduRTxTpApplicationPdu

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
PduRHandleId	1...1	Integer	**0** - 65535	Handle ID that is used by the application to access this PDU. The handle space must be contineous and zero based.
PduRPduRef	1...1	Reference	not specified	Reference to the global PDU. Establishes a link to the PduRRoutingPaths destination TP layer PDU. Please note that TP PDUs which have been referenced here must not be involved in any other routing path.

Table 6-54 Attributes of PduRTxTpApplicationPdu

6.5.1.22 VectorCommonData

Container Name	VectorCommonData
Path	\MICROSAR\PduR\VectorCommonData
Multiplicity	1...1
Description	

Table 6-55 Container VectorCommonData

Attribute Name	Multi-plicity	Value Type	Values Default value is typed bold	Description
BswmdVersion	1...1	String	**3.16.00**	Version of the BSWMD file that has been used to set up the ECU configuration

Table 6-56 Attributes of VectorCommonData

7 AUTOSAR Standard Compliance

7.1 Deviations

> The PDU Router ignores the return value of LIN IF transmission requests. This violates the PDU Router specification, however; there is no other way to overcome this problem without violating a SWS requirement.
The problem's origin is that LinIf_Transmit() always indicates an error in case of triggering the transmission of an unconditional LIN frame (LINIF341).
On the other hand, the PDU Router is designed in a way to not differentiate between the destination bus type when triggering the transmission of an I-PDU (PDUR255, PDUR256).

> Gateway Tp Routing:
If currently no buffer available, the PDU Router rejects a buffer request if the transport protocol (CAN, LIN or FR) requests a buffer. An automatic retry mechanism is not supported.

This means the functions PduR_<Lo>TpProvideRxBuffer return "BUFREQ_E_NOT_OK" instate of "BUFREQ_E_BUSY" in case of no buffer available currently. → Partial deviation of PDUR350 (Can), PDUR372 (Fr), PDUR394 (LIN).

Note
Counter measure: The Application has to configure a sufficient number of buffer elements (e.g. number of possible parallel routings) to avoid that all buffers are busy.

7.2 Limitations

> N:1 routing: routing of several source I-PDU to one or several destination I-PDUs is not supported for TP layer routing paths.

> M:N routing: mapping of several source PDUs to several destination PDUs is not supported.

> Routing between interface layer I-PDUs with different sizes on Rx and Tx side. If the destination I-PDU has a greater bit count than the source I-PDU it is not specified by AUTOSAR ([1]) how this should be handled. Therefore PDU Router currently does not support such routing paths. In case of the source I-PDU being larger than the destination I-PDU, solely the byte count of the destination I-PDU is routed.

> TP or interface layer routing (gateway use case) using a destination I-PDU that is also transmitted by an upper layer module (such as COM or DCM) is not supported.

> The TP layer gateway does not support the FrTp retry feature.

> If the gateway feature is used, the PduR configuration data must be linked to the default ROM section (must not declared as “far”).

> The data section used by PduR must be linked to the default RAM section (must not declared as “far”).

> Each interface layer routing relation can be assigned to a dedicated Tx Buffer. Sharing a Tx Buffer among several routing relations is not possible.

> When using the AUTOSAR 4.0 (draft) TP layer API, PduR supports CanTp and J1939TP TP interfaces only. Other TP layer modules are currently not supported by such deliveries.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
Application interface	API. An application interface is the prescribed method of a software component for using the available functionality. It should be distinguished between an application control interface and an application data interface.
Buffer	A buffer in a memory area normally in the RAM. It is an area, that the application has reserved for data storage.
Callback function	This is a function provided by an application. E.g. the CAN Driver calls a callback function to allow the application to control some action, to make decisions at runtime and to influence the work of the driver.
CANbedded	… is the trademark of the Vector communication stack.
Channel	A channel defines the assignment (1:1) between a physical communication interface and a physical layer on which different modules are connected to (either CAN or LIN). 1 channel consists of 1..X network(s).
Communication stack	The communication stack consists of the communication configuration and the communication kernel, a number of adaptive software components that cover the basic communication requirements in distributed automotive applications.
Component	CAN Driver, Network Management ... are software COMPONENTS in contrast to the expression module, which describes an ECU.
Confirmation	A service primitive defined in the ISO/OSI Reference Model (ISO 7498). With the service primitive 'confirmation' a service provider informs a service user about the result of a preceding service request of the service user. Notification by the CAN Driver on asynchronous successful transmission of a CAN message.
EAD	Embedded Architecture Designer; generation tool for MICROSAR components
Electronic Control Unit	Also known as ECU. Small embedded computer system consisting of at least one CPU and corresponding periphery which is placed in one housing.
Event	An exclusive signal which is assigned to a certain extended task. An event can be used to send a binary information to an extended task. The meaning of events is defined by the application. Any task can set an event for an extended task. The event can only be cleared by the task which is assigned to the event.
Gateway	A gateway is designed to enable communication between different bus systems, e.g. from CAN to LIN.
GENy	Generation tool for CANbedded and MICROSAR components
Indication	A service primitive defined in the ISO/OSI Reference Model (ISO 7498). With the service primitive 'indication' a service provider informs a service user about the occurrence of either an internal event or a service request

	issued by another service user. Notification of application in case of events in the Vector software components, e.g. an asynchronous reception of a CAN message.
Interrupt	Processor-specific event which can interrupt the execution of a current program section.
Interrupt service routine	The function used for direct processing of an interrupt.
Lower Layer	Components located in the Layer Model below the PDUR. See Figure 2-2 Interfaces to adjacent modules of the PDUR. These components are CAN, LIN, FR, MOST Interface or Transport Protocol and IPDUM.
Network	A network defines the assignment (1:N) between a logical communication grouping and a physical layer on which different modules are connected to (either CAN or LIN). One network consists of one channel, Y networks consists of 1..Z channel(s). We say network if we talk about more than one bus.
Network Management	The Network Managements manages the availability of different networks on a channel. It is responsible for the synchronized transition between the communication states 'sleep', 'prepare sleep' and 'active' for all modules. Network Management serves to ensure the safety and availability of the communications network of autonomous control units. The OSEK NM distinguishes between node-related (local) activities, e.g. initialization of the node and network-related (global) activities, e.g. coordination of global NM operating modes.
Overrun	Overwriting data in a memory with limited capacity, e.g. queued message object.
Post-build	This type of configuration is possible after building the software module or the ECU software. The software may either receive parameters of its configuration during the download of the complete ECU software resulting from the linkage of the code, or it may receive its configuration file that can be downloaded to the ECU separately, avoiding a re-compilation and re-build of the ECU software modules. In order to make the post-build time re-configuration possible, the re-configurable parameters shall be stored at a known memory location of ECU storage area.
Schedule table	Table containing the sequence of LIN message identifiers to be transmitted together with the message delay.
Scheduler	The algorithm which decides whether a task switch is to be affected and which triggers all necessary internal activities of the operating system is named scheduler. The scheduler decides whether a task switch is possible according to the implemented scheduling policy. The scheduler can be considered as a resource which can be occupied and released by tasks. Thus a task can reserve the scheduler to avoid task switch until the scheduler is released.
Signal	A signal is responsible for the logical transmission and reception of information depending on the restrictions of the physical layer. The definition of the signal contents is part of the database given by the vehicle manufacturer. Signals describe the significance of the individual data segments within a message. Typically bits, bytes or words are used for data segments but individual bit combinations are also possible. In the CAN data base, each data segment is assigned a symbolic name, a value range, a conversion formula and a physical unit, as well as a list of

	receiving nodes.
Transport Protocol	Some information that must be transferred over the CAN/LIN bus does not fit into individual message frames because the data length exceeds the maximum of 8 bytes. In this case, the sender must divide up the data into a number of messages. Additional information is necessary for the receiver to put the data together again.
Upper Layer	Components located in the Layer Model on the top of the PDUR. See Figure 2-2 Interfaces to adjacent modules of the PDUR. These components are Com, Dcm, ApplIf, ApplTp, IPDUM.
Use case	A model of the usage by the user of a system in order to realize a single functional feature of the system.

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BS	undefined
BSW	Basis Software
BSWMD	Basic Software Module Description
CAN	Controller Area Network protocol originally defined for use as a communication network for control applications in vehicles.
CF	TP Consecutive Frame
COM	Communication
CRC	Cyclic Redundancy Check
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
FC	TP Flow Control Frame
FF	TP First Frame
FIFO	First In First Out
FR	FlexRay Driver
GUI	Graphical User Interface
HIS	Hersteller Initiative Software
ID	Identifier (e.g. Identifier of a CAN message)
IF	Interface
I-PDU	Interaction PDU (Presentation Layer)
ISO	International Standardization Organization

ISR	Interrupt Service Routine
J1939	Is the vehicle bus standard used for communication and diagnostics among vehicle components, originally by the car and heavy duty truck industry in the United States.
J1939TP	J1939 Transport Layer
LIN	Local Interconnect Network
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
MOST	Media Oriented Systems Transport
NM	Generic Network Management Interface
OEM	Original Equipment Manufacturer
OSEK	Name of the overall project: Abbreviation of the German term "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" - Open Systems and the Corresponding Interfaces for Automotive Electronics.
PDU	Protocol Data Unit
PDUR	PDU Router
RAM	Random Access Memory
ROM	Read-Only Memory
RTE	Runtime Environment
SDU	Service Data Unit
SF	TP Single Frame
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
TMS320	Texas Instruments DSP
TP	Transport Protocol
XCP	undefined

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector.com